SEAMAIL

Newsletter from Software Engineers Association

Vol. 15, Number 1-3 October, 2006

目　次

MBT & SPIN 2006 参加報告	野中哲	1
SEA Forum June		5
開催案内		5
Firum Report	石川雅彦・櫻井麻里	6
スライド(オブジェクト指向の理想と現実)	酒匂寛	10
SEA Forum August		44
開催案内		44
SEA Forum「これからのシステム開発」報告	中谷多哉子	45
SEA Forum 始末	藤野晃延	53
スライド(Forum 概要)	中谷多哉子	62
スライド(ミスユースケース)	海谷治彦	66
スライド(超上流工程)	菊島靖弘	87
スライド(非正常系シナリオ)	三瀬敏朗	106
スライド(パネル討論の材料)	中谷多哉子	114
スライド(ビジネスモデリング方法論iMOY)	斎藤信也	120
スライド(要件を定量的に捉えた見積りモデル)	太田忠雄	126
参加者アンケートのまとめ		133
総会報告		138
編集後記		139

ソフトウェア技術者協会
Software Engineers Asociation

ソフトウェア技術者協会 (SEA) は，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が，そうした社会組織の壁を越えて，各自の経験や技術を自由に交流しあうための「場」として，1985年 12月に設立されました．

その主な活動は，機関誌 SEAMAIL の発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約 200人にすぎなかった会員数もその後増加し，現在，北は北海道から南は沖縄まで，350 余名を越えるメンバーを擁するにいたりました．法人賛助会員も 17社を数えます．支部は，東京以外に，関西，横浜，名古屋，九州，広島，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

「現在のソフトウェア界における最大の課題は，技術移転の促進である」といわれています．これまでわが国には，そのための適切な社会的メカニズムが欠けていたように思われます．SEA は，そうした欠落を補うべく，これからますます活発な活動を展開して行きたいと考えています．いままで日本にはなかったこの新しいプロフェッショナル・ソサイエティの発展のために，ぜひとも，あなたのお力を貸してください．

ＳＥＡＭＡＩＬ Ｖｏｌ．１５，Ｎｏ．１－３ ２００６年１１月１日発行 編集人 岸田 孝一
発行人 ソフトウェア技術者協会（ＳＥＡ）
〒１６０-０００４ 東京都新宿区四谷３－１２ 丸正ビル５Ｆ
T: 03-3356-1077 F: 03-3356-1072 E-mail: sea@sea.or.jp URL: http://www.sea.jp/
印刷所 市田印刷株式会社 〒１１４-００１４ 東京都北区田端２－３－２５
定価 １，５００円 （禁無断転載）

MBT,SPIN2006参加報告

野中　哲
(有）トゥルーロジック
September 13, 2006

1. はじめに

2006年3月、オーストリアのウィーン工科大学で開催されたワークショップMBT2006(Model Based Testing 2006)とSPIN2006に参加してきた。 もともとモデル検査ツールSPINのワークショップSPIN2006にのみ参加する予定で 日程を組んでいたのだが、同時期にModel Based Tesingというタイトルの ワークショップが組まれていることを知り、 フォーマルな仕様記述とテストの関係に多少興味を持っていた事もあり、せっかくはるばるウィーンまで出かけるのだからと、こちらのワークショップも覗いて見ることにした。 この2つののワークショップはETAPS2006 (Joint European Conferences on Theory and Practice of Software) の併設ワークショップとして企画されている。そもそも このETAPSという会議自体、小さなワークショップの集合体としてデザインされているようで、 単体では参加者の確保が難しそうな、やや特殊で狭い分野をテーマとするワークショップが並んでいる。

2. MBT2006

MBTは第2回目の開催である。3/25,26の二日間にわたり開催された。参加者の数はおよそ50名程であろうか。
1日目の最初は招待講演でGoogle 社のHarry Robinson 氏が、"Model-based Testing for the Masses" という演題で話をした。 Harry さんの話は、大部分の聴衆がアカデミーの 世界の人たちだったこともあり、具体的なデータを使いながら

- テストには、なかなか予算が付かずツールも満足に買えない。フリーのツールに期待している。
- 人材が少ない。他の分野を専攻した人が多く、専門にソフトウエアを勉強した人は少ない。 オートマトン理論や正規言語の事を知っている人は皆無。
- よって、Zによる仕様なんぞ敷居が高過ぎて現場には無理
- でもSpec# はお勧めできる
- 論文をもっとテスト関係のジャーナルに書いて欲しい。
- 学生には、テストがエキサイティングで挑戦しがいのある分野であることを教え、もっと人材を送り込んで欲しい。

といった内容だった。
講演終了後、彼を捕まえ少し話をしたが、気さくなカリフォルニア人という印象の人であった。 日本へくる予定は無いかと聞いたところ、 「とりあえず日本へ行く予定は無い。でもいつでもメールくれ！」言ってくれた。 何かの機会

に彼を日本に呼び、話をしてもらうと面白いかもしれない。
2日目の招待講演は、IBM Israel, Haifa Research LaboratoryのAlan Hartman 氏で演題は"Ten years of model based testing - A Sober Evaluation"というも の。10年間にわたるモデルベーステストの経験談を話してくれた。 話の骨子は、モデルベーステストを実践しても、それがなかなか現場に根付かない という体験の紹介だった。確かに効果はある、良い論文も書けた。しかし、その 手法がその後継続的に現場で実践されるという状況になっていない、という嘆き。 フォーマルアプローチに全般に関してよく聞く話だなという印象を持った。
一般の論文発表で面白かったのがBernhard K. Aichernig と Chris George.らに よる "When Specification-based Testing Fails" というもの。「仕様書をベースに テストするのは原理的に無理である」というような哲学的内容だが、前提となる知識が 広範囲に必要な上に、現世利益はあまりありそうもないので、詳細な内容の追求は、老後の 楽しみにとっておこうと思う。
全般的なこのワークショップの印象は、テストを謳っているだけに、フォーマルアプローチに関連した会議の 中では現場・実用志向が強いように思えた。しかしながら、「モデルベーステスト」という用語について どのような技術を指すのか明確なコンセンサスが形成されていないようにも思える。 「モデルベーステスト」が体系的な技術の分野として整理、確立されるにはもう少し時間が必要な ようだ。

3. SPIN2006

SPIN2006は、毎年開催されているモデル検査ツールSPINに関するワークショップである。 この会議はもう16回目だが、私が参加するのは昨年のサンフランシスコに続き2回目 になる。基本的なアルゴリズムの話等が多く、私が興味を持っている、産業界への 応用に関する話題が少ない点にやや難があるのだが、それでも時々、今後の参考に なりそうな面白い話が聞けるので、参加するようにしている。
今回のワークショップでの中心テーマは"Directed Model Checking"というキーワードに 集約されそうだ。これはモデル検査で問題となる状態爆発に対して、発見的アルゴリズム やAIの手法を応用して、探索空間を狭め効率良く捜そうという手法。いずれも研究段階の論文で 私の興味を引くものではなかった。
実プロジェクトへの応用としては、招待講演の「インテルの事例発表」が面白かった。 モデル検査は、そもそもハードウエアの検証としてまず普及したのであるが、 その話を裏付ける、ペンティアムプロセッサの開発で、どのようにモデル検査が 用いられているかに関する話であった。CPUの開発は、

- 初期投資が大きく、回収までの期間が長い
- 一旦市場に出た後、不具合が見つかると、回収のコストが非常に高くつく
- 製品が複雑で、全ての検査を網羅的に行うことは実質的に不可能

などの理由により、モデル検査やフォーマルアプローチを導入する動機 が大きい。ハードウエアの検証はソフトウエアの検証に比べ、簡単だと思われている

が それは誤り。消費電力、占有面積、スピードなど諸々の制約により、 細かな最適化が行われており、論理的に素直な設計になっているとは限らず、 これら職人芸的な設計の検証を行うのは難しい作業である。フォーマルアプローチの 最大のセールスポイントは100% のカバレージである。証明を行うにせよ モデル検査を行うにせよ、全ての場合を網羅することが出来る。これは他のテスト手法には ない特徴である。フォーマルアプローチは、従来のテスト手法を置き換えるものではなく、 補完するものであり、この100% を網羅するという特徴を、いままでのテスト手法と 組み合わせて品質向上を図ることが大切。 といった内容であった。次に面白かったのが、

"Verification of Medical Guidelines by Model Checking – A Case Study" Simon Baumler, Michael Balser, Andriy Dunets, Wolfgang Reif, Jonathan Schmitt (Univ. of Augsburg, D)

という発表。 これはドイツの医療ガイドラインをモデル検査したという話。まずベースとして 自然言語による医療のガイドラインの文書が存在していているのだが、これをエキスパート システム等に応用するため、医療ドメイン専用ののフォーマルな言語に書き換える。 そして、そのフォーマルな記述の中から、医療行為のプロセスに関する部分ををモデル検査した。 医療行為は、並行プロセスの集合と考えられるので、モデル検査と相性が良い。 検証する性質としては、例えば「治療は必ず終了する」とか「計測したデータは 必ず使われる」とかいったものになる。世の中には法律や規則のようなものが沢山あるが、 これらを検証し、矛盾や欠陥がないことを調べるというのは面白い仕事かもしれない。 後に知ったのだが、こういった分野は Regulation Verificationと呼ばれているようだ。

4. まとめ

ワークショップは全て終了。総勢参加者は600名くらいだそうで、こん なにフォーマル関係の人が沢山いるとは驚きである。欧州のフォーマル派の層の厚さを感じる。 産業界の発表は少ないが、今回はインテルの動きが目立った。レセプ ションをスポンサーしたり、コンパイラ開発とかチップの検証といった分野で活発に求人もしてた。

シュプリンガーの出店にはビヨルナー先生の最新の著作"Software Engineering"が3巻とも並んでいた。 重たそうだったので、買おうかどうか悩んだが 割引の上、日本まで無料で送ってくれるというので3冊とも買ってしまった。 実はその後、紆余曲折があり、結局現物が無事日本に届くまでには2ヶ月を要したのだが...

5. おまけ：ウィーン観光

中心部の建物はどれも立派である。特に天井の高さに驚かされる。 帝国時代の富と権力を誇示するに充分な威容である。 発表の合間の息抜きにドナウ川見物に出かけた。 ウィーンのダウンタウンの近くを流れているドナウ川は、実は本流ではなく 市内に引き込んだ人工的な運河である。そのため、たいへん小じん

まりとしている。 どうしても本流を見てみたくなり、地下鉄で郊外まで出かけてみた。ついでにその先にある 国連のビルも見学。核開発に対する査察で有名なIAEAなどの入ったビルを外から見学。内部を見学できるガイドつきツアーもあるのだが、これは時間が合わず断念。 ドナウ川の本流は幅広く、水量も多く立派な河であった。雪解け水のせいかどうか 分からないが、水は粘土色に濁っている。ヨハンシュトラウスの曲のにあるような 「美しき青きドナウ」は、実在したのか想像上の産物なのか少し考えてしまった。

ウィーンといえば、やはり映画「第三の男」である。出発前に 500円でDVDを購入。行きがけの飛行機の中で、じっくりと予習をした。 映画に出てきた名所は一応 行ってみないと気がすまない。あの有名な観覧車にのり、カフェモーツアルトで コーヒーを飲む。ラストシーンの墓地は、空港行きの電車の中から遠目に眺めた。 もう一つ、下水道を見学したかったのだが、これはかなわず。

いままでの人生でオペラなぞ見たことが無かったが、昼間オペラ座の前をぶらぶらしていると ダフ屋のおじさんが近づいてきて、今夜のチケットがあるという。天井桟敷（6階）の最前列で 90ユーロを提示されたが、そんなにお金を持っていないというと、立ち見はどうだ？ と聞かれる。三時間を立ち見は辛いので、椅子に座ってみたい、と言うと 同じく天井桟敷の最後列（この後ろが立見席になる）が50ユーロ。めでたく商談成立。 渡されたチケットが本物なのかどうか心配だったが、ちゃんと入場できた。 演目は「ドン・ジョバンニ」。座席の前に小さな液晶パネルがあって、これに台詞が 出てくる。英語かドイツ語を選べるようになっている。しかし、この字幕を追いかけて いると、全く舞台がみれないので、開始後すぐに台詞を追いかけるのあきらめた。

別の日の昼休みには、ウィーンフィルの新年コンサートで有名な「楽友協会ホール」 を見学。ガイドツアーに参加。案内の人は流暢なドイツ語と英語で 解説してくれた。木造で、非常に音が良く響くホールで、確かにここで オーケストラを聴いてみたいと思わせてくれるホールだった。

ウィーン滞在中は全く忘れていたのだが、思い起こせばここは「論理実証主義派」 (ウィーン学団）の総本山であった。普通のガイドブックには、そのような事は書いていない ので気が付かなかった。今年は生誕150年とかで、街中はモーツァルト一色であったし、 少し歩けば、やたらに音楽家の銅像はあるのだが、よく捜せばウィトゲンシュタインや ゲーデルの銅像もどこかにあったのかもしれない。 次回訪問のチャンスがあったら、このあたりも押さえておこうと考えている。

SEA Forum June

オブジェクト指向の理想と現実

主催: ソフトウェア技術者協会 (SEA)

オブジェクト指向は、多くのソフトウェア開発プロジェクトにおいて日常的に適用される開発手法となっています。多くの場合、適用の目的は生成したコンポーネントの再利用率を高めることにより、高生産性、高保守性を狙うことにあると考えられます。この傾向は、オブジェクト指向分析/設計/開発を効率的に実施する技法とツールの普及によって益々拍車がかかっています。

しかし普及と成熟の一方で、誤解や混乱が拡大しているとの指摘も挙がっています。例えば、開発フェーズにおいてオブジェクト指向言語を採用しているものの、一つのクラスしか存在しないプログラム、1メソッドが数百行のコードから成るプログラム、責務の混合したプログラム、相互に依存度の高いコンポーネント群などの存在を容易に許してしまいます。このような問題は、プロジェクト管理や品質管理体制の強化によっては解決することが困難な側面があります。つまり、分析設計開発の各フェーズにおいて支援ツールの採用実績を挙げ、テストを通過し、納期が守られているというポイントはクリアしているため、見かけ上はオブジェクト指向開発プロジェクトは成功したと評価されますが、実際は手続き型言語と変わらない設計、開発が行われているケースも観測されています。

オブジェクト指向の理想と現実の乖離に焦点をあてた今回のフォーラムでは、黎明期よりオブジェクト指向手法に親しみ、理論家であると同時に優れた実践家としても知られるお二人の講師をお迎えしました。この場で私達は、オブジェクト指向が正しく適用された世界がどのようなものであるかを再認識すると同時に、実際の開発現場においてその世界に到達するにはどうしたらよいかをディスカッションしたいと思います。

実際の開発プロジェクトにたずさわる、多くのソフトウェア技術者の参加をお待ちしています。

********** 開 催 要 領 **********

1. 日時:2006 年 6 月 16 日(金) 13:30 受付, 14:00 開始, 17:30 終了
2. プログラム

13:30 - 14:00 受付

14:00 - 16:00 講演 (1) オブジェクト指向において保存・変形・視座・多重・名前・位相を

青木淳(SRA 先端技術研究所)

オブジェクト指向プログラミングを20年以上も続けてきた観点か ら、保存・変形・視座・多重・名前・位相について言い及びます。現在、日経ソフトウエアに連載されている「青木淳のプログラマ魂」のトピックスになります。それぞれについて、どのようにオブジクト指向で昇華されているのかを開示してゆきます。一介のプログ ラマが到達したひとつの世界観(理想と現実)の紹介になります。

(2) オブジェクト指向の理想と現実(「使えない」モデルはいらない)

酒匂寛(Designers' Den)

オブジェクト指向開発においては「モデル」が話題になることが多いのですが、現場におけるそのモデルの位置付けは曖昧です。多くの場合初期段階にスケッチのように書かれて、あとは保守も されないという場合が多いのではないでしょうか。しかし、こうなる理由は明らかで、書かれたモデルが「検証」できない ことに起因しています。多くの開発プロジェクトの混乱が仕様(モデル)の混乱に起因することを考えると、これは大変残念なことです。今回は形式手法をプロジェクトに導入することによって、モデルを活性化する方法について、事例を交えながらお話します。

16:00 - 16:10 Break

16:10 - 17:30 討論:オブジェクト指向の理想と現実 コーディネータ: 桜井麻里(SEA 幹事)

パネリスト:: 上記の講師のお2人

3. 会場: 全国情報サービス産業厚生年金基金(JJK)会館 7F B 会議室 (東京都中央区築地 4-1-14)
4. 定員: 50名 (申込み順, 定員になり次第、受付を締め切ります)

SEA Forum June
オブジェクト指向の理想と現実
報告

石川雅彦
桜井麻里

SEA Forum「オブジェクト指向の理想と現実」は、6月16日金曜日、東京東銀座のJJK会館で開催されました。この時の様子を、開催案内に記載した文書をはさみながら紹介していきたいと思います。

開催趣旨

まず、このフォーラムは誰のためのフォーラムかといいますと、開発現場で実際に開発に携わっているエンジニアのためのフォーラムとして企画しました。実際の開発現場で役立つためには、格好のいい言葉だけでは間に合いません。しかし、本音のフォーラムというものはともすれば、現状肯定や愚痴の披露となる恐れがあります。まさに理想と現実で、フォーラムではこのバランスを保つことが難しいと思われました。考えた挙句、フォーラムの開催趣旨説明を兼ねた開催案内を以下のように始めました。

オブジェクト指向は、多くのソフトウェア開発プロジェクトにおいて日常的に適用される開発手法となっています。多くの場合、適用の目的は生成したコンポーネントの再利用率を高めることにより、高生産性、高保守性を狙うことにあると考えられます。この傾向は、オブジェクト指向分析/設計/開発を効率的に実施する技法とツーの普及によって益々拍車がかかっています。

しかし普及と成熟の一方で、誤解や混乱が拡大しているとの指摘も挙がっています。例えば、開発フェーズにおいてオブジェクト指向言語を採用しているものの、一つのクラスしか存在しないプログラム、1メソッドが数百行のコードから成るプログラム、責務の混合したプログラム、相互に依存度の高いコンポーネント群などの存在を容易に許してしまいます。このような問題は、プロジェクト管理や品質管理体制の強化によっては解決することが困難な側面があります。つまり、分析設計開発の各フェーズにおいて支援ツールの採用実績を挙げ、テストを通過し、納期が守られているというポイントはクリアしているため、見かけ上はオブジェクト指向開発プロジェクトは成功したと評価されますが、実際は手続き型言語と変わらない設計、開発が行われているケースも観測されています。

オブジェクト指向の理想と現実の乖離に焦点をあてた今回のフォーラムでは、黎明期よりオブジェクト指向手法に親しみ、理論家であると同時に優れた実践家としても知られるお二人の講師

をお迎えしました。この場で私達は、オブジェクト指向が正しく適用された世界がどのようなものであるかを再認識すると同時に、実際の開発現場においてその世界に到達するにはどうしたらよいかをディスカッションしたいと思います。

オブジェクト指向において保存・変形・視座・多重・名前・位相を。

まず、最初は SRA 先端技術研究所の青木淳さんの講演です。このタイトルは、青木さんが、日経ソフトウエアに連載している「青木淳のプログラマ魂」のトピックスを連結したものということです。それぞれのトピックについて、どのようにオジェクト指向で昇華されているのかを開示し、　あるプログラマが到達したひとつの世界観（理想と現実）の紹介を行なう、というものでした。

プレゼンのスクリーンには PC の画面が映し出され、そこにはあるムービーの画面が現れていました。ムービー画面に対して「動け」と指示するとムービーが動き始めます。まるで、その画面と直接対話するかのようにして操作しています。

引き続いて、3 次元グラフィックマルチメディアライブラリ「じゅん」を例にとってオブジェクト指向を示します。　スクリーンには様々な形の図形が現れています。青木さんはそれらの図形に対して演算を試みます。この形とこの形を「足す」とか「引く」という演算です。

私はそのプレゼンを聞きながら、図形同士の演算が何を意味しているのか考えてみました。私はグラフィックを取り扱う専門家ではありませんので、青木さんと同様な操作に熟練する必要はありません。しかし、エンジニアであれば何らかの分野の熟練者であるはずで、その熟練した分野で使用する基本的なプリミティブについては、自由に「演算」できなければいけないのではないか、という考えに至りました。

オブジェクト指向の理想と現実（「使えない」モデルはいらない）

少しの休憩に引き続いて次は SEA 幹事であり、Designers' Den 社の酒匂寛さんの講演です。開催案内に寄せた文章で酒匂さんは以下のように述べています。

「オブジェクト指向開発においては「モデル」が話題になることが多いのですが、現場におけるそのモデルの位置付けは曖昧です。多くの場合初期段階にスケッチのように書かれて、あとは保守もされないという場合が多いのではないでしょうか。」。そして酒匂さんはこの理由を、「書かれたモデルが「検証」できないこと」に起因していると看過しています。また、「多くの開発プロジェクトの混乱が仕様（モデル）の混乱に起因する」とも述べています。

事態を収拾するための仕組みとして酒匂さんが提示したものは、

「契約による設計」。システム開発の 3 つの視点-課題・仕様・設計。

でした。

まず、「契約による設計」では、モジュール間の関係を「契約」とみなします。そして「契約」の記述とは、権利と義務を定義することと述べます。権利と義務を「事前条件」「事後条件」で記述することをすすめています。
この「契約による設計」はオブジェクト指向だけに特有な概念ではないということです。

酒匂さんがプレゼンを行なった「課題・仕様・設計 extreme」で私の印象に残ったのは、顧客要件を関係者がどのように誤解したかを示した寓話のようなイラストでした。
野原に一本の太い木が立っています。その太い枝にロープで 3 段ほどの板がゆわえられています。どうやらそれはブランコのようです。これが顧客が欲しがっているもののようです。しかし、プロジェクトリーダーはそれを 1 段のブランコだと誤解します。更にアナリストが顧客要件に基づいたデザインでは、太い木が上下に分断され、その間に1段のブランコが載っているという、美しいが「ありえない」設計になっています。一方、営業は顧客に対して行なった約束は、ひじかけ付き重役椅子でつくったブランコを提供するというものでした。イラストの多くを省略しますが、実は顧客は木にぶらさがった古タイヤのブランコがほしかっただけ、という落ちがついています。ここでは利用者、開発者の視点がいかに異なるかが端的に表されています。システム開発には、利用者の視点（課題）、開発者の視点(設計)、そして利用者と開発者の共有する視点である仕様があり、システム開発には3つの成果物がある。そして、課題・仕様・設計の関係を図で視覚的に解き明かしてくれました。

ディスカッション

お二人の全くタイプの異なる発表を受け、まずは、お二人の発表の内容に対して会場の参加者から質問を受け付けました。続いて、発表の内容に限らず、お二人に聞きたい質問を広く会場から受け付け、お二人に答えていただく形で、ディスカッションのきっかけとしました。

最初の質問は、「青木さんの発表の中で、"Object Oriented" の "Object" を『もの』と訳したところがそもそもの間違いだった、というような発言がありましたが、『もの』でないとしたら、"Object" って何ですか？ 」というものでした。

この質問に対して、青木さんからは、中国語での表現が適切であるように思うとして、その紹介がありました。「もの」というよりは「こと」という感じのようです。

最後の質問は、「お二人にとって、オブジェクト指向の現実って何ですか？」というものだったと記憶しています。お二人の発表自体が、お二人にとっての「現実」だったと思うのですが、会場の参加者にとっては、「理想」として映ったようです。

最後に

案内には、「オブジェクト指向が正しく適用された世界がどのようなものであるかを再認

識すると同時に、実際の開発現場においてその世界に到達するにはどうしたらよいかをディスカッションしたい」と書きましたが、実際には、ディスカッションの時間が、お二人への質問コーナーで終わってしまった感は否めません。

それでも、お二人の発表を通じて、「オブジェクト指向が正しく適用された世界がどのようなものであるかを再認識」でき、「オブジェクト指向の理想と現実」に関して、ほんの僅かな時間ではありますが、一緒に考えることはできたものと思います。

積極的に質問をして下さった多くの参加者と、それぞれ熱心に答えてくださった青木氏と酒匂氏に、感謝します。ありがとうございました。

なお、このフォーラムが行われた６月は、まだ、日経ソフトウエアの青木さんの連載が始まっていない時期だったこともあり、著作権者の要請により、青木さんの発表内容を印刷して配布することが叶いませんでした。現在では、この連載は終了しています。興味のある方は、日経ソフトウエアに連載された青木さんの記事をご参照ください。

以上

オブジェクト指向の理想と現実

「使えない」モデルはいらない

酒匂寛（Designer's Den)

本日の話題

- 契約による設計のススメ
  - モジュールの義務と権利
- 課題・仕様・設計
  - 開発プロセスにおけるモデルの位置付け
- 仕様から設計へ
  - 記述のフレームワーク
- 形式仕様とその効果
  - 形式仕様記述の役割
- おまけ：記述の基礎

契約による設計のススメ

契約による設計とは

- モジュールとモジュールの関係を「契約」とみなして設計を行う手法
- あるモジュールの仕様記述は「契約書」とみなすことができる

権利と義務の表明

- プログラムの正しさを議論する際にしばしば以下のような表記が用いられる

$$\{P\}\ A\ \{Q\}$$

- この表記の意味は「条件 P が成立しているときに、Aが実行されると条件Qが成立する」という意味である
- すなわちPはAにとって、呼び出される際に要求できる権利であり、QはAにとって果たすべき義務ということになる
- Pがきちんと成立しているかどうかを気にしなければいけないのはAのクライアント(呼び出す側)であり、A自身には責任はない

2004/05/21
5
(C) 2004, Sako Hiroshi

契約の記述

- ではソフトウェアモジュール間の契約をどのように記述すべきだろうか？
- 契約とはそれに参加する「甲、乙」の義務と権利を定義するものである
  - 甲にとっての義務は乙にとっての権利
  - 甲にとっての権利は乙にとっての義務
- 権利と義務を「事前条件」「事後条件」で記述する。多くの場合「使われる側」のモジュールが「契約書」を用意する

2004/05/21
6
(C) 2004, Sako Hiroshi

契約例

- Class 印刷キュー
  - Method 印刷要求登録（r : 印刷要求）
    - 事前条件
      - not Self.isFull ()
    - 事後条件
      - Self.queueLength == Self.queueLenghth @ pre + 1
      - Self.lastEntry() == r
- Class 商品取引
  - Method 商品注文(s: 注文伝票) : 注文
    - 事前条件
      - 商品チェック(s.商品) and 顧客チェック(s.顧客)
    - 事後条件
      - Result.対応注文(s) and 注文リスト.includes(Result)

UML の中での位置付け

- CASE ツールでは、操作の仕様定義の中に、事前事後条件の指定を許すものが多い
- 事前条件、事後条件の記述方法は自然言語でも構わないが、UML の中では OCL（Object Constraint Language: オブジェクト制約言語）の利用が推奨されている
- （注）しかし UML/OCL はいまや巨大で中途半端
  - VDM++ などの形式仕様記述言語で、足りない部分を補い、より簡潔で「検証可能な」記述を目指すべき
  - 現場での導入が一部始まっている

「契約による設計」は オブジェクト指向や UML だけに特有な概念ではない

- 普通の手続き型言語を用いても、モジュール間の契約をはっきりと記述できれば、責任の所在をはっきりさせることが、より簡単になる
  - 設計が局所的になり、変更に強くなることが期待される
  - 障害発生時に問題を「契約の破綻」として認識できる場合が多い

2004/05/21
9

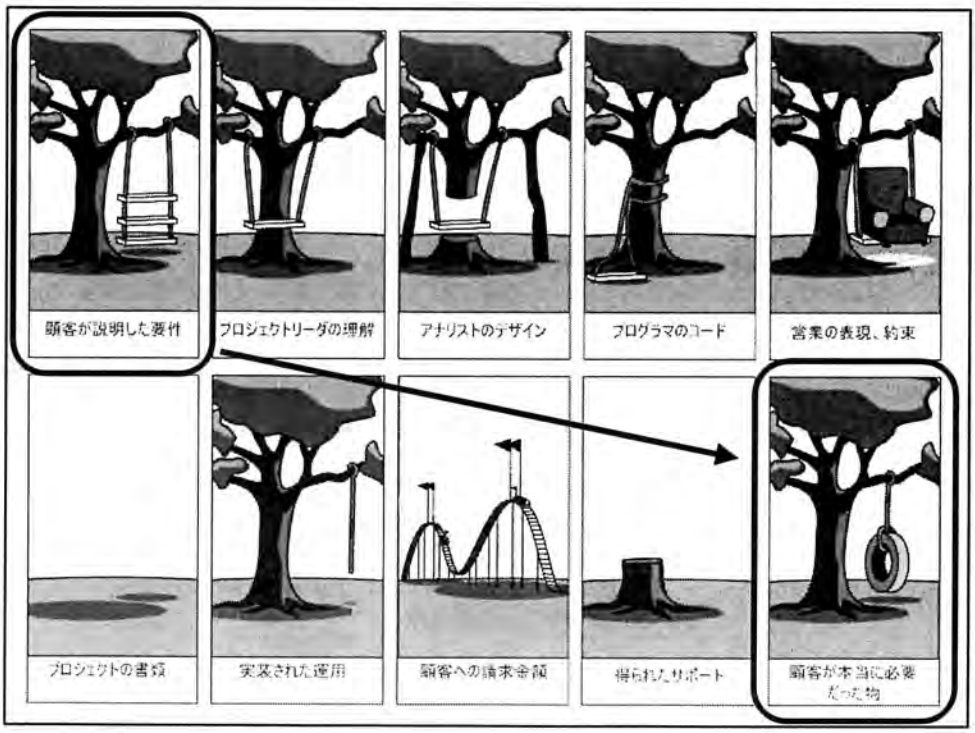
「契約による設計」:むりやりのまとめ

- 契約による設計を理解することは、クラスの仕様を記述する上で重要な意味をもっている
  - 理解性も向上する(もちろん「簡潔」かつ「正確」に書いた場合に限る)
- といいながらも
  - 契約による設計を直接表現できる言語は少ない
  - UML には契約を記入するための場所は用意されているものの、CASE ツールによる支援は原始的なものに留まっている
- こうした不利な条件を考えても、なお
  - 契約による設計を意識することが、責任の明確化、局所化を促し、堅牢なモジュール作りの手助けとなる

2004.05.21
10

課題・仕様・設計
extreme

酒匂寛@DDC
sakoh@ba2.so-net.ne.jp



ネタ本 - 1

課題・仕様・設計 不幸なシステム開発を救うシンプルな法則
著者: 酒匂 寛著

本体価格: ¥2,200
出版:インプレスネットビジネスカンパニー
サイズ:A5判 / 207p
ISBN:4-8443-1866-7
発行年月:2003.12

課題: 何を解決したいのか
仕様: その課題をどのような仕掛けで解決するのか
設計: その仕掛けをどのように最適な形で実現するのか

2004/05/21
13
(C) 2004, Sako Hiroshi

システム開発の3つの視点

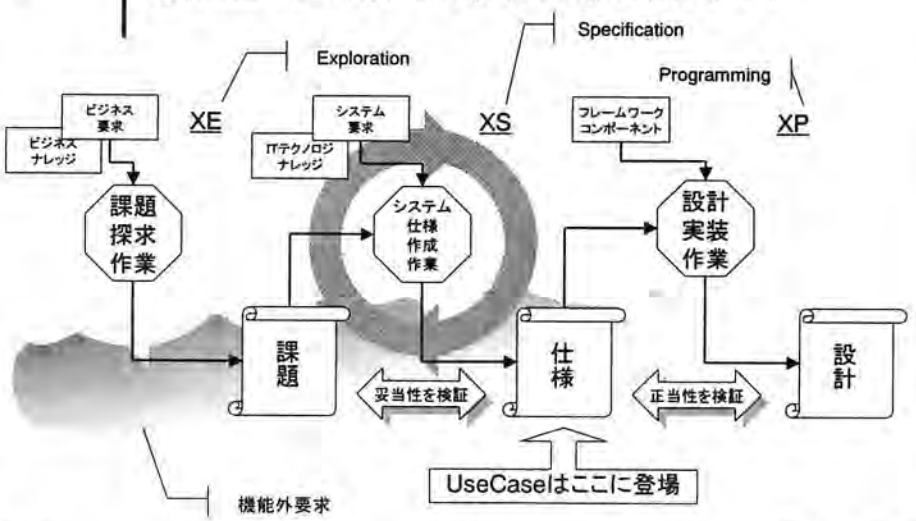
- 課題
  - 利用者の視点
    - 現実世界にどのような問題を抱えており、そのうちどの部分をシステム化して解決したいと思っているのか
  - 入力:ビジネス要求
- 仕様
  - 利用者と開発者の視点
    - 上で挙げられた課題の解決を、どのようなシステムで支援するのか
  - 入力:課題＋システム要求
- 設計
  - 開発者の視点
    - 要求されるシステムの仕様をどのように設計すべきか
  - 入力:仕様＋最新構築技術

2004/05/21
14
(C) 2004, Sako Hiroshi

システム開発3つの成果物

- システムを構築していくためには次の3つの成果物が必要です
  - 課題定義
    - 現状分析と改善項目の抽出
      - 解決すべき課題は何か、問題領域の規則は何かを記述します
    - 新ビジネスモデルの提示
      - 抽出されたビジネスモデルを最適化し、あるべき姿を提示します
      - 計算機による実現を特に意識しません。個別のアプリケーション(外部へのアダプテーション)もここでは詳細には定義しません。もちろんメモを残すことは常に可能です
  - システム仕様定義
    - IT技術を用いてどのようなシステムを構築すればよいかを定義します
      - 新しいビジネスモデルを実現するために提供すべき機能
      - 課題を解決するために用意すべき機能
    - アプリケーションやデバイスなどへの役割分担はここで行われます
  - 設計実装定義
    - システム仕様に基き、実際に動作するシステムを構築(設計・実装)します。成果物はシステムそのものです

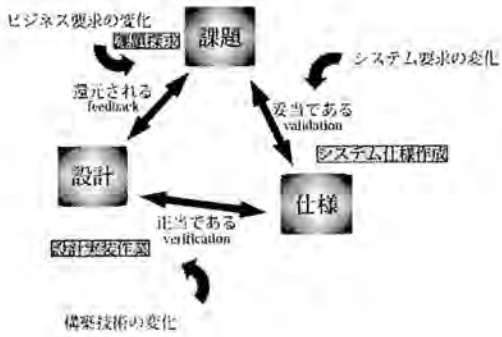
課題-仕様-設計の関係(1)



成果物と作業の関係：追記

- 前図は作業と成果物の一般的な関係を表しているだけで、その作業手順を強制しようとするものではありません
  - 課題探索、仕様作成、設計実装はそれぞれの入力を元に並行に作業することが可能です
- 例えばウォーターフォール型の開発プロセスだけに適用されるものではないということに注意して下さい。各作業は並行に進めることができますし、成果物も決して上流から順に作成されることを意味していません。要するに矢印は「影響」を表しているだけです
  - UP のような繰り返し型開発や、XP のような方法論等何を持ってきても前図に示した関係は崩れません

課題-仕様-設計の関係（2）



各レベルへ展開される「改善」

課題	受付済み患者の呼び出しミスを回避（長時間の待ち回避を含む）
システム仕様	患者追跡ユースケースアプリケーションが、患者一覧画面の中で、受付後一定時間以上（10分以上、20分以上）状態の変化していない患者が存在することに注意を促す
設計実装	患者追跡アプリケーションの患者一覧画面で、各患者の行の背景色を以下のように設定する。 受付直後：緑色 受付10分以降：黄色 受付20分以降：赤色 診察受付済：白色

2004/05/21
19
(C) 2004, Sako Hiroshi

memo:「事実」と「願望」の区別

- ときに問題文書の中に、「領域の性質を述べている」(=事実)のか、あるいは「これから領域内でそのような性質を満たして欲しい」(=願望)のかが不明な記述が登場する場合がある
- 同じ課題を記述した文書の中に、両者が入り混じって記述されていると、仕様を作成する側は正しいシステム仕様を作成することができない

2004/05/21
20
(C) 2004, Sako Hiroshi

課題の構成要素

- 目的
  - 利用者の業務と解決すべき課題を記述
- 構成
  - 改善項目
    - 既存の業務(手動、自動を問わず)に対する改善要求
      - この項目への対応は以降、業務モデル、システム仕様、設計実装の各段階に反映される
  - 現行モデル
    - 業務イベント
    - 業務ルール
  - 業務モデル
    - 業務プロセスモデル
      - 業務を構成するイベント(問題領域分析から導出)に対応
        - 業務オブジェクトに対する操作として表現
    - 業務オブジェクトモデル
      - 業務プロセスモデルを可能にする情報モデル
        - UML 表記等による
    - 業務ルール
      - 制約、計算式、判定条件その他上記両者(オブジェクト、プロセス)に適用される様々なルール

2004/05/21

21

仕様の構成要素

- 目的
  - 構築するシステムの機能仕様、機能外仕様を記述
- 構成
  - アクタ・プロセス対応表
    - アクタと業務プロセスの対応関係を定義
  - 業務文脈モデル
    - 課題を解決するために、何処で誰がいつ何を行うかを定義するモデル
      - アクタ、業務プロセス、インターフェイスの配置を考える
      - 例:Who-Where-When-What Diagram (4WD)
  - ユースケースアプリケーション(UCAP)
    - 業務論理の組み合わせによる、ドメインモデルの操作体系
    - 業務プロセスは業務論理に対応、システム化の都合により業務論理の追加も可能
  - ドメインモデル
    - 課題解決のために必要な詳細な情報モデル
  - 業務論理
    - 情報モデルを操作するための業務論理。UCAPから使われる

2004/05/21

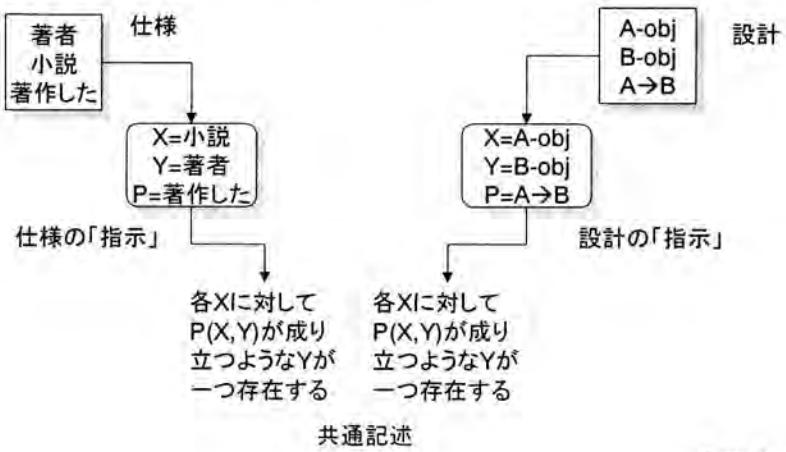
22

設計の構成要素

- 目的
  - システム仕様の最適配置と実現
- 構成
  - システムアーキテクチャ
  - ソフトウェアアーキテクチャ
  - ミドルウェア
  - フレームワーク
  - コンポーネント
  - アプリケーション
  - etc..
- とはいうものの。。。。
  - 業務仕様と設計がどこで出会うかを明確な形で表現できるようにしたい → フレームワークが必要
  - 例：汎用事務処理フレームワーク Gofo

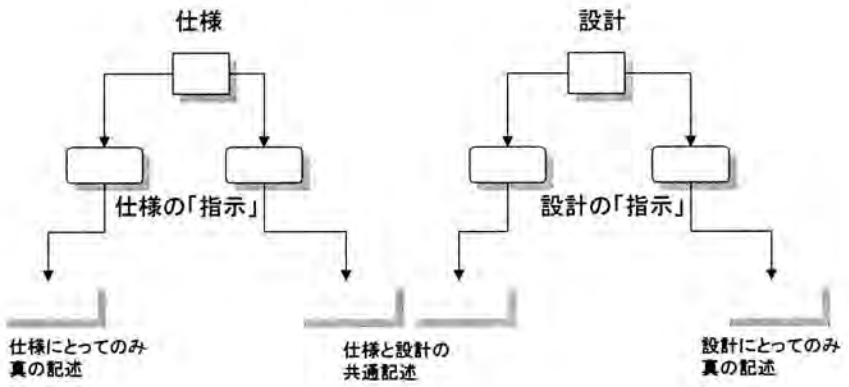
2004/05/21
23
(C) 2004, Sako Hiroshi

仕様と設計「記述」の接点



2004/05/21
24
(C) 2004, Sako Hiroshi

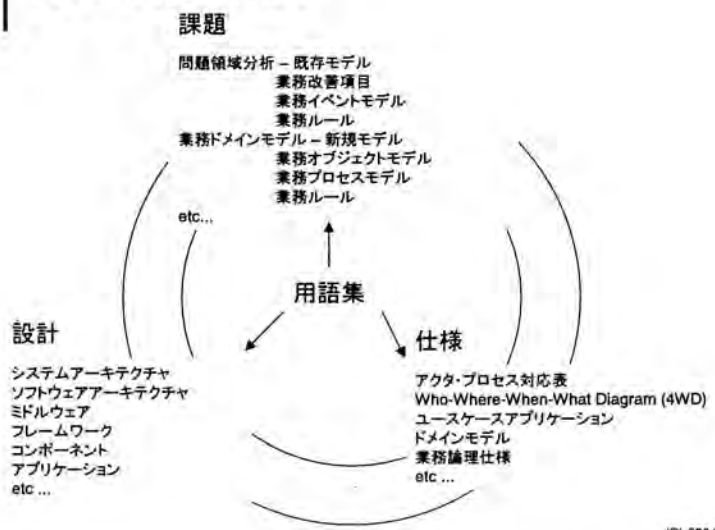
「記述」の重なり



2004/05/21

25

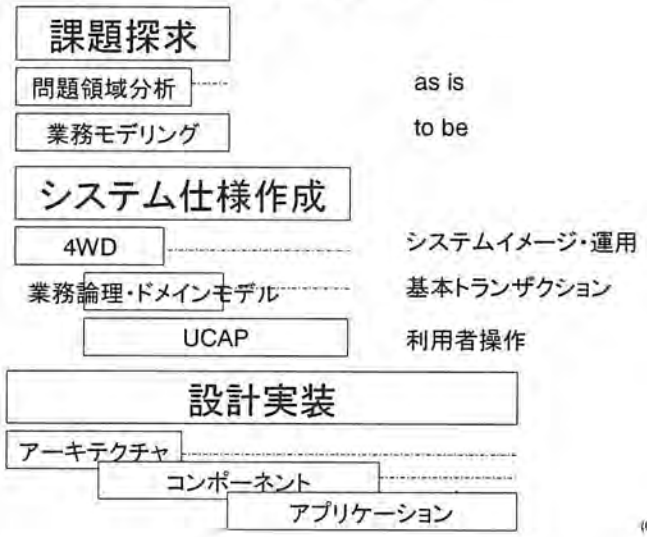
成果物の例



2004/05/21

26

並行して進む作業



2004/05/21
27

(C) 2004, Sako Hiroshi

仕様記述～設計のための フレームワーク

汎用事務処理フレームワークGofoの例

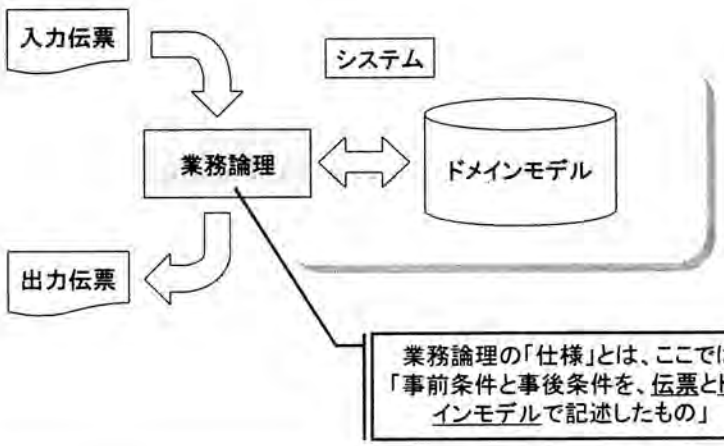
Gofo = good old fashioned office

Gofo の二面性

- 仕様記述のための Gofo
  - 業務を実現するためのシステム仕様記述
- 設計実装のための Gofo
  - 仕様記述された結果をなるべく素直に実装するための仕掛け

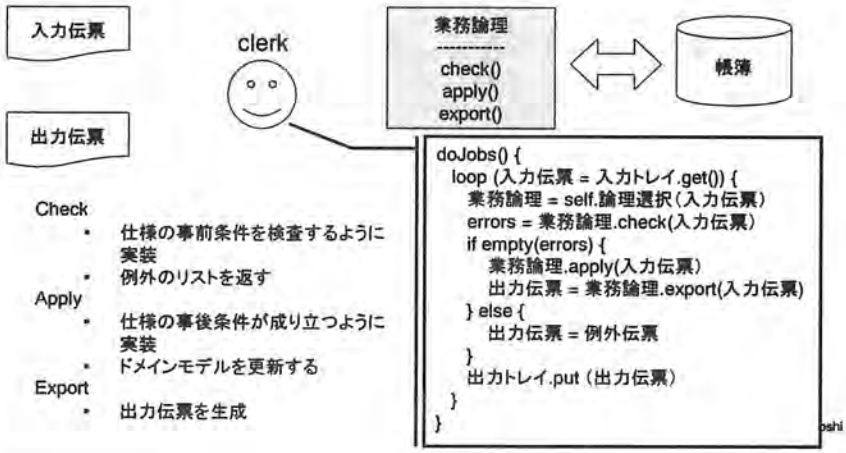
2004/05/21
29
(C) 2004, Sako Hiroshi

仕様記述のための Gofo



2004/05/21
30
o Hiroshi

設計実装のための Gofo（仕様との関係）



Check
- 仕様の事前条件を検査するように実装
- 例外のリストを返す

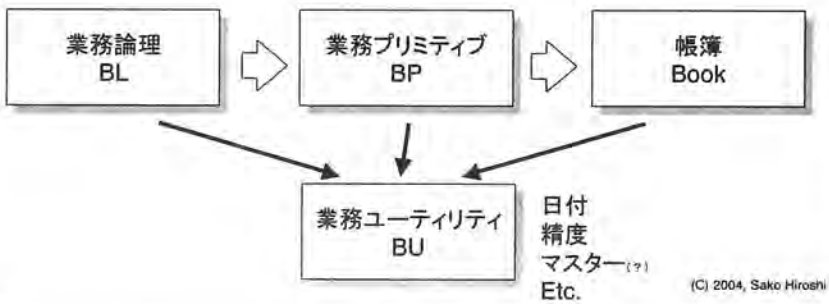
Apply
- 仕様の事後条件が成り立つように実装
- ドメインモデルを更新する

Export
- 出力伝票を生成

2004/05/21
31

設計実装のための Gofo（内部構成要素）

業務論理が直接帳簿を参照、更新する設計実装を考えても良いが、再利用性保守性ともに悪くなる。
このため、中を責務によってもう少し構造化する



2004/05/21
32
(C) 2004, Sako Hiroshi

設計実装のための Gofo（各要素の位置付け）

業務論理 BL	ユースケースの実装に必要となる、業務の基本単位。業務論理は「業務プリミティブ」を組み合わせて実現する。情報構造の変化に対応するため、自ら帳簿に直接アクセスしない
業務プリミティブ BP	業務を構成する基本的なサービスを提供する。例えば顧客、口座、注文など
帳簿 Book	ドメインモデルから導かれる情報構造を、二次元の表として表現したもの。集合と明細の両方を扱うインターフェイスを提供する。RDB で実現しやすいが、実装は RDB に縛られない。一帳簿＝一テーブルという強い縛りがあるわけでもない。

2004/05/21

33

設計実装のための Gofo（トランザクション）


```
doJobs() {
  loop (入力伝票 = 入力トレイ.get()) {
    業務論理 = self.論理選択(入力伝票)
    errors = 業務論理.check(入力伝票)
    if empty(errors) {
      業務論理.apply(入力伝票)
      出力伝票 = 業務論理.export(入力伝票)
    } else {
      出力伝票 = 例外伝票
    }
    出力トレイ.put (出力伝票)
  }
}
```

ポリシーの指定方法は
パラメータもしくはサブクラス
（未定）

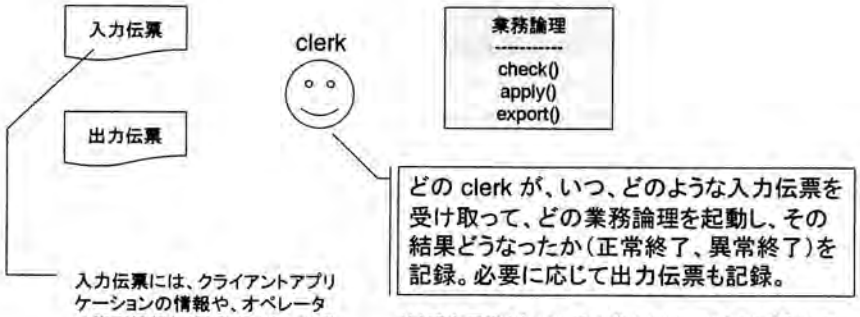
トランザクション境界をどこにするかは clerk のポリシーに依存する
- 伝票一枚ごと
- 全伝票処理後

トランザクションを制御するのは原則として clerk

2004/05/21

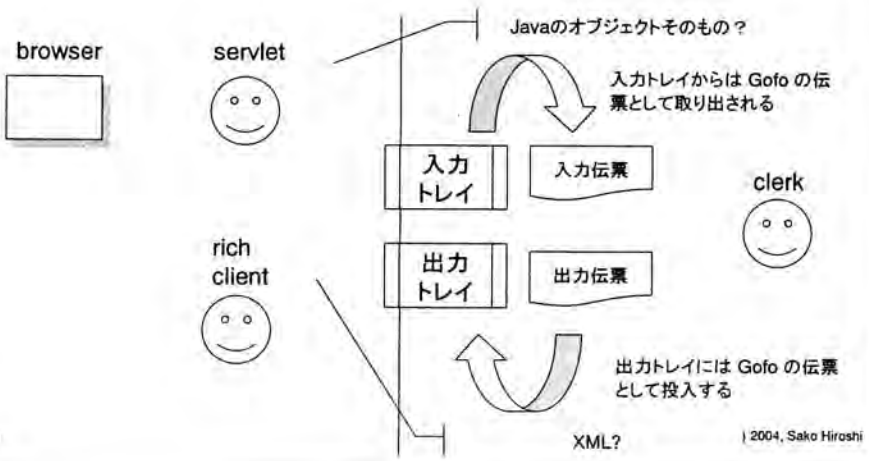
34

設計実装のための Gofo（業務履歴）



注）現在用意されているのは SlipBinder という仕掛けのみ。SlipBinder は伝票刺のイメージ。現在 RDB を使って実装しているが、単なるファイルでも良いかも（目的による。要検討）

設計実装のための Gofo（対クライアント）



2004/05/21
36
| 2004, Sako Hiroshi

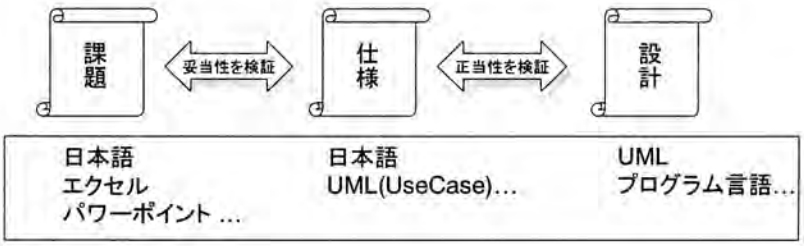
形式仕様記述の導入と効果

Extreme Specification Programming

2004/05/21
37
(C) 2004, Sako Hiroshi

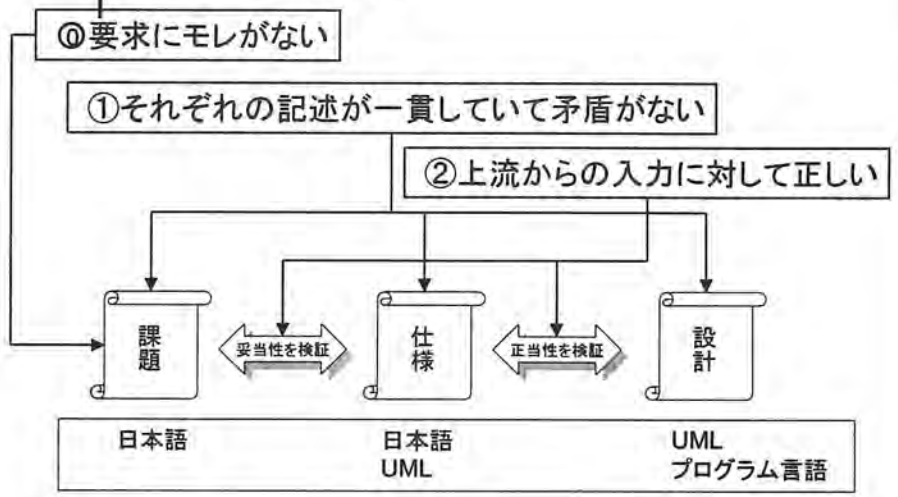
現在の記述のメディア

例えば・・・



2004/05/21
38
(C) 2004, Sako Hiroshi

検証の視点



2004/05/21
39
(C) 2004, Sako Hiroshi

検証の手段

- 0:要求にモレがない
  - 欠けている要素を見つける
    - 自動化は困難で問題領域の専門家の知恵を集めるしかない
- 1:それぞれの記述が一貫していて矛盾がない
  - 記述内部の矛盾を見つける
    - 文法的に正しく意味的に矛盾していないかどうか
- 2:上流からの入力に対して正しい
  - 記述「間」の整合性を調べる
    - 基本的に上流の記述からテストケースを導き、下流の記述をテストする

2004/05/21
40
(C) 2004, Sako Hiroshi

形式手法を用いた記述

- 前頁のような検証を「効果的」に行うためには、検証対象の「記述自身」が曖昧さのない形で表現されていた方がよい
- 形式手法とは数学的な基礎を下敷きにしながら、記述対象の「厳密なモデル」を記述し、質の高い検証を進めていく手段である

2004/05/21
41
(C) 2004, Sako Hiroshi

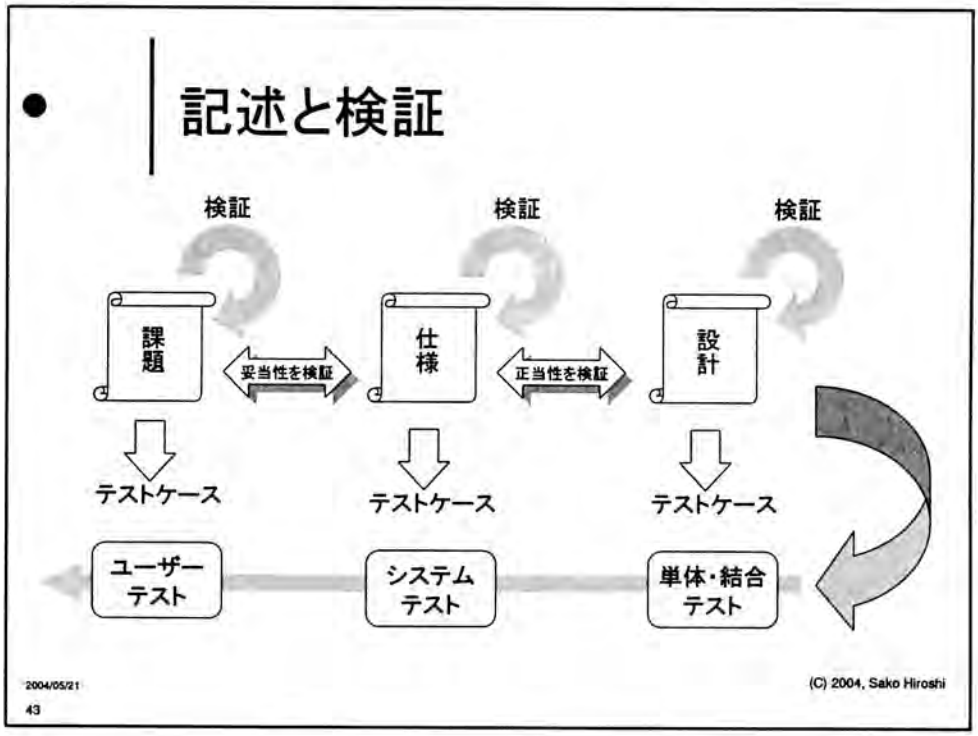
形式記述/モデルのご利益

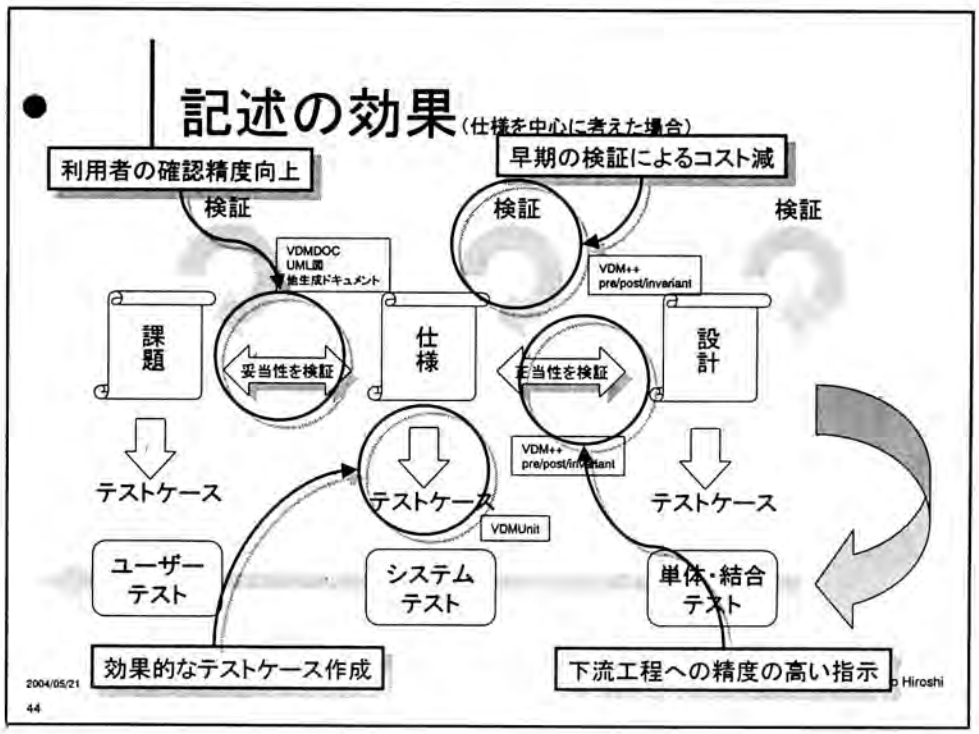
形式記述/モデル

人間に対する
より抽象的で理解しやすい記述の提示

機械処理による検証

2004/05/21
42
(C) 2004, Sako Hiroshi


記述と検証
検証
検証
検証
課題
仕様
設計
妥当性を検証
正当性を検証
テストケース
テストケース
テストケース
ユーザーテスト
システムテスト
単体・結合テスト
2004/05/21
43
(C) 2004, Sako Hiroshi


記述の効果（仕様を中心に考えた場合）
利用者の確認精度向上
早期の検証によるコスト減
検証
検証
検証
VDMDOC
UML図
自生成ドキュメント
VDM++
pre/post/invariant
課題
仕様
設計
妥当性を検証
正当性を検証
VDM++
pre/post/invariant
テストケース
テストケース
テストケース
VDMUnit
ユーザーテスト
システムテスト
単体・結合テスト
効果的なテストケース作成
下流工程への精度の高い指示
2004/05/21
44

形式仕様記述言語の導入

- ここまで見てきたように、正確な仕様の記述が検証を容易にする
  - こうした目的のために開発されてきたのが形式仕様記述言語でありVDM++もその代表の一つである
- VDM++とは
  - 汎用の仕様記述言語
    - 形式手法の道具
    - モデル指向の仕様記述言語

2004/05/21
45

VDM++の沿革

- VDM
  - ウィーン開発手法(Vienna Development Method)
    - IBMのウィーン研究所で最初に開発
    - VDM-SLはISO標準(1996年)
- VDM++
  - 欧州連合ESPRIT計画のAFRODITEプロジェクト産
- VDMTools
  - CSKのVDM-SL,VDM++開発ツール群

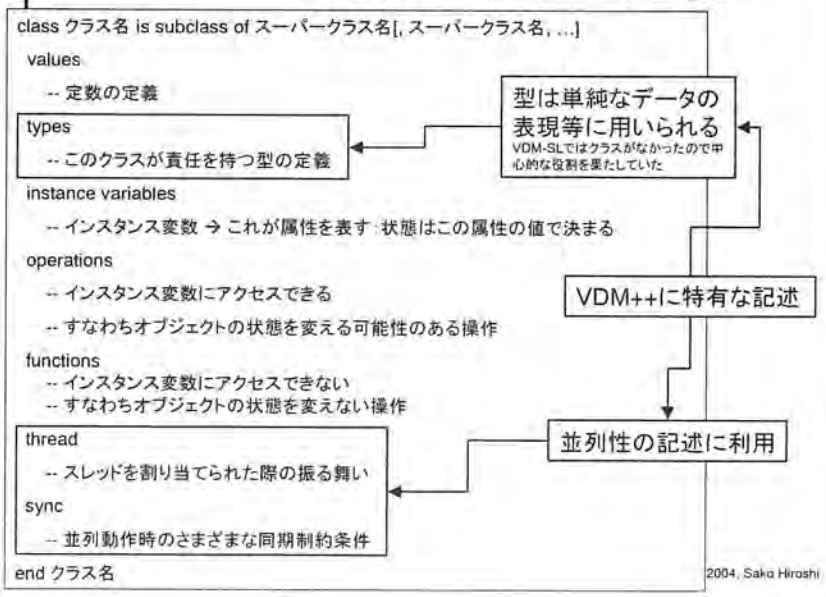
2004.05.21
46

VDMの教科書

- VDM-SL
  - J. フィッツジェラルド、P.G. ラーセン他 著
    ソフトウェア開発のモデル化手法、岩波書店、2003
  - 荒木啓二郎、張漢明 著
    プログラム仕様記述論、オーム社、2002
- VDM++
  - J. Fitzgerald, P.G. Larsen他 著
    Validated Design for Object-Oriented Systems, 2005
    - http://www.vdmbook.com/
  - CLIFF B JONES, SYSTEMATIC SOFTWARE DEVELOPMENT USING VDM
    SECOND EDITION, Prentice Hall International,1990
    - ftp://ftp.ncl.ac.uk/pub/users/ncbj/ssdvdm.ps.gz

2004/05/21
47
(C) 2004, Sako Hiroshi

VDM++ の特徴 - クラス定義



2004/05/21
48
2004, Sako Hiroshi

VDM＋＋の特徴 - 不変条件

types
-- このクラスが責任を持つ型の定義

```
Day = nat1
inv d == d <= 31;
Month = nat1
inv m == m <= 12;
Year = int;
```

Year, Month, Day 型を宣言している

それぞれの型には必要なら不変条件を与えることができる

instance variables
-- インスタンス変数 → これが属性を表す：状態はこの属性の値で決まる

```
day : Day;
month : Month;
year : Year;
inv if month in set { 4, 9, 6, 11 }
    then day <= 30
    else (month = 2) => (day <= 29)
```

Year, Month, Day 型の属性（インスタンス変数）を宣言している

複数の属性間の制約を不変条件という形で表すことができる

2004/05/21

49

VDM＋＋の特徴 – 事前/事後条件

operations
-- インスタンス変数にアクセスできる
-- すなわちオブジェクトの状態を変える可能性のある操作

```
預金(金額 : int) 新残高 : int
ext wr 残高 : int
pre 金額 >= 0
post 残高 = 残高~ + 金額 and 残高 = 新残高;
```

いずれも「陰仕様」という形式で定義されている。この形式は「どのようにするか」ではなく「何をするか」を記述するためのものである

pre: 事前条件

post: 事後条件

がそれぞれ定義されている。

functions
-- インスタンス変数にアクセスできない
-- すなわちオブジェクトの状態を変えない操作

```
税込み価格(金額 : int)  r : real
post r = 金額 * 1.05;
```

事前条件は呼出す側の「義務」を表し、事後条件が呼ばれた側の「責任」（おこなうべきこと）を表している

2004/05/21

50

VDM++が仕様記述に提供するもの

- オブジェクト指向の採用による「仕様」のカプセル化と情報隠蔽
- 不変条件による型の制約や、状態の制約
- 事前条件、事後条件による操作/関数に対する義務と責務
- インタプリタによる仕様の「実行」

2004/05/21
51
(C) 2004, Sako Hiroshi

memo:記述の違い(Java vs VDM++)

```
import java.util.*;
class「プラント」{
   Map 勤務予定;
   // ...
   Set 勤務予定を確認する(Integer 専門家) {
    TreeSet result = new TreeSet();
    Set keys = 勤務予定.keySet();
    Iterator iterator = keys.iterator();
    while (iterator.hasNext()) {
     Object 期間 = iterator.next();
     if (((Set) 勤務予定.get(期間)).contains(専門家))
       result.add(期間);
     }
     return result;
   }
}
```

<Java>

VDM++ には集合、系列、写像などが言語に組み込まれているため記述がコンパクトになる。他にパターンマッチングも強力
Java の TreeSet などは実装上の概念である

```
public 勤務予定を確認する : 「専門家」 ==> set of【期間】
勤務予定を確認する(専門家) ==
    return { 期間 | 期間 in set dom 勤務予定 &
                    専門家 in set 勤務予定(期間) };
```

<VDM++>

2004.05.21
52
(C) 2004, Sako Hiroshi

事例1 – 証券業務

- (旧)日本フィッツ社での経験 - 証券業務
  - Gofo/VDM++で仕様記述
  - Gofo/C++, Gofo/Java で設計、実装
- 特徴
  - 仕様記述のフレームワークと設計・実装のフレームワークを用意して、両者の間の連携をスムースに
  - 仕様レベルでのテストと、テストケースの流用

2004/05/21
53

事例2 – 非接触型ICカード

- 某非接触型 IC カード開発プロジェクト
  - API の仕様を VDM++ で記述(フレームワーク化)、すべてのマスターに
  - VDM++ の記述と C による実装で多くのテストケースを共有
- 特徴
  - 仕様レベルのテスト
    - 実装(ハードウェア)が登場する以前に、仕様を検証しながら策定
    - 実装のテストとテストケースを共有
  - 記述の明確化によるコミュニケーションの向上
  - 詳細仕様書の生成

2004 05 21
54

「記述」の「基礎」

現実世界とシステムをつなぐもの

2004/05/21
55
(C) 2004, Sako Hiroshi

ネタ本 - 2

ソフトウェア要求と仕様 — 実践、原理、偏見の辞典
マイケル ジャクソン (著), 玉井 哲雄 (訳), 酒匂 寛 (訳)

本体価格：¥3,570
単行本: 267 p ; サイズ(cm): 21 x 15
出版社: 新紀元社 ; ISBN: 4775302876 ; (2004/04)

ソフトウェア開発を「記述」の体系として捉え、それにまつわる様々な話題を取り扱った本

2004/05/21
56
(C) 2004, Sako Hiroshi

どのような「記述」にも ...

- 以下の要素が含まれる
  - 「指示」を用いた基礎用語
  - 「指示」を組み合わせて作られる「定義」
  - 対象の制約について述べる、「定義」を用いた「論理式」
- 以下のスライドでそれらを説明します

2004/05/21
57
(C) 2004, Sako Hiroshi

指示 : Designations - 1

- 問題領域の規則や性質は、問題領域の言葉(「課題」の言葉)で定義されなければならない
- 問題領域で認識されて区別される「現象」には、「指示」を使って名前をつけることができる
- 「指示」は認識規則と指示された用語の組み合わせの一覧である → 用語集と呼んでも良い
- 「指示」は現象を記述するために存在する。それ自身なんらかの真偽を記述するものではない

2004 05 21
58
(C) 2004, Sako Hiroshi

指示:Designation - 2

指示の例

x は期間 y における会社の社員である = Employee(x, y)
x は会社の歴史内のある期間である = Intvl(x)
x は期間 z における y の上司である = Manager(x, y, z)

2004/05/21
59
(C) 2004, Sako Hiroshi

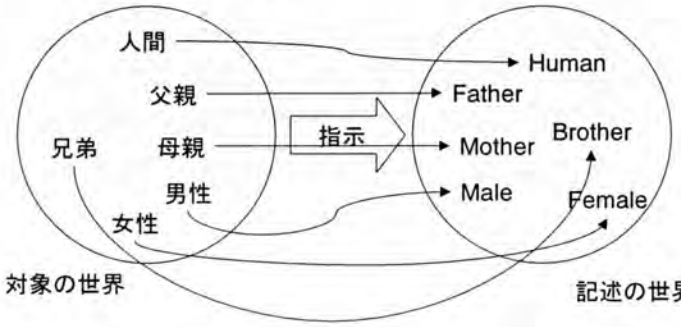
定義 : Definitions - 1

- 記述の中で用いられる用語はどこかで定義されなければならない。そうでなければ読む人はそれが何について述べているのかを知ることができない
  - 同様に書き手もいざというときに確信を持つことができなくなる
- 用語を定義するための一番簡単な方法は「指示」を定義することである
  - x は人間 = Human(x)
  - x は男性 = Male(x)
  - x は女性 = Female(x)
  - x は y の遺伝的母親 = Mother(x, y)
  - x は y の遺伝的父親 = Father(x, y)
- この調子でどんどん「指示」を増やしていくこともできるが...
  - x は遺伝的な y の兄弟 = Brother(x, y)
  - x は遺伝的な y の姉妹 = Sister(x, y)
- しかし、このやりかたはあまりふさわしいやりかたとはいえない

2004 05 21
60
(C) 2004, Sako Hiroshi

定義:Definitions - 2

- 何が問題かといえば、指示が多くなればなるほど、認識すべき世界と、それを記述した世界の関係が「濃く」なりすぎるということ
- 対象の世界と記述の世界との間にはなるべく「狭い橋」を架けなければならない



2004/05/21
61
(C) 2004, Sako Hiroshi

定義:Definitions - 3

<u>指示を増やす代わりに、指示を組み合わせた定義を行う</u>

x は遺伝的な y の兄弟 = Brother(x, y)

Brother (x, y) ≡ Male(x)
∧ ∃f･(Father(f, x) ∧ Father(f, y))
∧ ∃m･(Mother(m, x) ∧ Mother(m, y))

2004/05/21
62
(C) 2004, Sako Hiroshi

反駁可能な記述

- 開発文書内に現れる、どのような記述も「反駁可能な」ものである必要がある
  - 「反駁可能である」とは、「間違っている」という意味ではなく、反駁を受けた際に「その真偽が曖昧なく判断できる」という意味である
- ある記述を見たときに「...いや、それは予約という言葉で何を意味するかによるよ」というような議論になるようでは、その記述は反駁可能とはいえない
  - ソフトウェア開発において、反駁可能な記述を得るためにはなによりも明確な「指示」が必要である

2004/05/21
63
(C) 2004, Sako Hiroshi

述語論理 - 1

述語論理は事実を一般化して表現することができる(課題や仕様の規則を表現するために強力な武器となる)
例えば

知っている (x, y)

という述語が「x は y を知っている」という意味だとすると、x=花子, y=Java とすることによって

花子はJavaを知っている

という事実を表すことができる
この述語に∀(すべて),∃(存在する),∃!(ただ一つ存在する)といった限量子を組み合わせて使うことによって、より一般的な事実/規則を表すことができる

2004/05/21
64
(C) 2004, Sako Hiroshi

述語論理 – 2：曖昧な記述

- 「誰でも休日が好きだ」の意味は？
  - 誰でも全ての休日が好きだ。つまり誰でも仕事が休みになれば嬉しいという意味
  - 誰でも好きな休日が一つある。つまり誰でもある特定の休日 – 例えば大晦日 – が好きだという意味
  - 誰でも自分が好きな休日が少なくとも一つはあるという意味。例えば太郎は正月が好きで、花子は海の日と緑の日が好きといったこと
  - 誰でもちょうど一つだけ好きな休日があるという意味。例えば太郎は正月だけが好きで、花子は建国記念日だけが好きというような場合

2004/05/21
65
(C) 2004, Sako Hiroshi

述語論理 - 3
述語論理による曖昧な記述の除去

- 以下のような述語を用意する
  - 人(x) .. x は人である
  - 休日 (x) .. x は休日である
  - 好き(x, y) .. x は y が好きだ
- このとき以下の記述を見てみよう
  - 誰でも全ての休日が好きだ。つまり誰でも仕事が休みになれば嬉しいという意味
  - $\forall x, y \cdot ((人(x) \wedge 休日(y)) \rightarrow 好き(x, y))$
- もし「誰でも休日が好きだ」という代わりに、術後論理を用いて、曖昧さのない記述で書かれていれば、この部分に関する誤解は激減する

2004/05/21
66
(C) 2004, Sako Hiroshi

述語論理 - 4
述語論理による曖昧な記述の除去

- 誰でも好きな休日が一つある。つまり誰でもある特定の休日 – 例えば大晦日 – が好きだという意味
  - ヨ x・(休日(x)∧(∀y・(人(y)→好きだ(y, x))))
- 誰でも自分が好きな休日が少なくとも一つはあるという意味。例えば太郎は正月が好きで、花子は海の日と緑の日が好きといったこと
  - ∀x・(人(x)→ヨy・(休日(y)∧好きだ(x, y)))

2004/05/21
67
(C) 2004, Sako Hiroshi

まとめ:「記述」の構成要素

- どのような形式にせよ含まれるのは。。。
  - 「指示」を用いた基礎用語
  - 「指示」を組み合わせて作られる「定義」
  - 対象の制約について述べる、「定義」を用いた「論理式」
- 質のよい記述とは
  - 限られた数の意味の明快な「指示」
  - 豊富な「定義」
  - 豊富な「論理式」

2004/05/21
68
(C) 2004, Sako Hiroshi

SEA Forum August

これからのシステム開発:要求工学の挑戦

主催: ソフトウェア技術者協会(**SEA**)

要求工学の適用範囲は,設計制約の獲得と定義までを含めると,実装プロセスに先行するすべてのプロセスを網羅する技術分野ということになります.最近のシステムが実世界に与える影響は,情報システム,組み込みシステムにかかわらず,無視できないほど重大かつ重要なものになりつつあります.われわれソフトウェア開発技術者は,要求定義の重要さ,顧客を巻き込んだシステム開発の必要性を再認識し,新たなプロセス改善を目指すべき時期に来ているのではないでしょうか.

このフォーラムでは,要求獲得,要求定義,要求の仕様化,要求管理に関するいくつかの講演およびパネルディスカッションを通じて,フォーラムに参加される方々に以下のような共通認識をもっていただこうと考えています.

1)要求定義の重要性,要求獲得/定義の技術が存在すること (ミスユースケース,ゴール指向分析など)
2)現実世界とシステム世界の境界を見つめ直す必要があること (超上流工程,ビジネスモデリング)
3)要求プロセスと開発プロセスとの統合化が必要であること (民生品の信頼性・安全性向上のための要求定義)
4)要求の見積もり技術が必要であり,成果を出していること (ジャステック法)

スピーカとしては,要求工学の新進気鋭の研究者として数々の研究実績を持たれている信州大学の海谷治彦氏,超上流工程の提唱者としてSECリサーチフェローでもあるアイネスの菊島靖弘氏,家電製品などの民生品の信頼性向上,安全性向上のための非正常要求を獲得する手法としてESIM法を開発されている松下電工の三瀬敏朗氏をお招きし,それぞれの手法の解説をしていただきす.パネルディスカッションでは,上記3名の講師の他に,顧客のプロジェクトへの貢献度を定量化する手法を含む見積もり技術を適用されているジャステックの太田忠雄氏,要求獲得に先駆けた開発者と顧客との関係構築とビジネスモデリングを目指す手法 MOYA を適用されている NTT データの斎藤信也氏にも御登壇願って,日本における要求工学の強みと弱み,課題への取り組みなどについて議論します.

要求獲得,要件定義,顧客との関係改善に関心をお持ちのみなさんのご参加をお待ちします.

********** 開 催 要 領 **********

1. 日時:8月21日(月) 13:00~18:00
2. プログラム(予定)
   13:00 - 13:30 受付
   13:30 - 14:15 講演: "ミスユースケース,セキュリティ要求" 海谷治彦(信州大)
   14:15 - 15:00 講演: "超上流工程" 菊島靖弘(SECリサーチフェロー,アイネス)
   15:00 - 15:45 講演: "非正常系シナリオ" 三瀬敏朗(松下電工)
   15:45 - 16:00 Coffee Break
   16:00 - 18:00 討論:「これからのシステム開発:要求工学の挑戦」
      コーディネータ: 中谷多哉子(筑波大学)
      パネリスト: 上記講師3人
      斎藤信也(NTT データ))
      太田忠雄氏(ジャステック)
3. 会場: 全国情報サービス産業厚生年金基金(JJK)会館 7F B 会議室 (東京都中央区築地 4-1-14)
   http://www.jjk.or.jp/info/guidemap.html
4. 定員: 50名 (申込み順,定員になり次第,受付を締め切ります)

SEA Forum August：これからのシステム開発

要求工学の挑戦報告

筑波大学大学院&（有）エス・ラグーン
中谷多哉子

1．はじめに

要求工学の適用範囲は，設計制約の獲得と定義までを含めると，実装プロセスに先行するすべてのプロセスを網羅する技術分野ということになる．最近のシステムが実世界に与える影響は，情報システム，組み込みシステムにかかわらず，無視できないほど重大かつ重要なものになりつつある．要求定義が問題だ，要求プロセスに課題があると口で言うのは簡単だが，実際にどれほど我々がこのプロセスや技術について議論してきただろうか．その問題は広く，深いためか，ほとんど議論されてこなかったというのが現状であると思う．

要求工学への関心を高める活動は，情報処理学会のソフトウェア工学研究会の中で 8 年前から行われてきた．最初は，要求工学ワーキンググループという形で 1998 年に始められた．当初は，日本における研究者の育成が主たる目的であったが，2004 年の要求工学国際会議（RE'04）が京都で開催されることが決まった年あたりからチュートリアルを定期的に開催するなど，一般の関心を励起させる啓蒙活動も始めた．今回のフォーラムでご講演いただいた海谷先生は，要求工学ワーキンググループの主査を務めてもいる．今年は，要求工学の研究者，実務者を集め，問題の共有と更なる啓蒙活動を積極的に行うべく，ソフトウェアシンポジウム 2006 でワークショップを開催するなど，公的な活動を活発化した．

本稿では，7 月に熊本で開催されたソフトウェアシンポジウム 2006 のワークショップ報告を最初に行い，続いて 8 月に開催された SEA Forum の報告を行う．

2．グループH：要求工学ワークショップ（ソフトウェアシンポジウム 2006）

今年のソフトウェアシンポジウムでは，伝統的に行われている論文発表セッションとは別にワークショップが開催された．開催されたワークショップは 8 つであったが，要求工学ワークショップがグループHであったことからもわかるように，ワークショップの一覧を見て，重要な分野が欠けていることに気づき，締め切り間近に参加することを決めた．開催目的は，勿論，要求工学が現実世界とシステムとの境界領域を担う重要な分野であることをソフトウェアシンポジウムで討論することであった．参加希望者は 12 名であったが，業務の都合により実際の参加者は 11 名となった．

個々の発表内容をここで触れる余裕はないが，図 1 に全体の討論内容を総括した認知マ

ップを紹介する．図には，社会学的アプローチから工学的アプローチからを網羅する軸を横に，縦には問題領域理解から要求獲得を経て仕様化へ至る軸を設け，参加者の発表内容をプロットしてある．この図の軸にはメモリがあるわけではないので，直観的なマップである．また，参考までに，既存の主たる要求工学に関係する手法もマップした．これらが，ある程度の指標となるであろう．第三象限は空欄であるが，ここにはミスユースケース，シナリオ分析，クレーム分析など，システムの仕様に基づいて，システムが社会に与える影響を分析する手法をマップすることができる．

図 1　要求工学ワークショップで得られた認知マップ

ワークショップでは，要求工学に関する問題意識を持った実務家や研究者が集ったわけであるが，総じて，システムと社会との関わりを分析しなければ，現在の開発プロセスが抱えている問題を解決することは困難であるという点で共感することができた．第一回目の SEA 主催の要求工学ワークショップとしては，上々の成果であったと思っている．すなわち，

1）**我々がシステムに求める「もの」は，現実世界では，どのような「こと」と関係しているのか．**
2）**システムが果たすべき責務は，開発者と要求者とで，どのように共有していけばよいのか．**
3）**システムは，完成後に，どのような悪影響を社会に与えてはならないのか．**
4）**技術者教育では，何を教えるべきか．**

に，今後の要求工学が解決すべき課題があるようである．

3．SEA Forum August 報告

フォーラムの参加者募集記事で「要求工学の挑戦」と銘打ったためか，参加者アンケートを読むと多少の誤解を生じた感があった．実際のところ，意味のある要求を正確に，簡単に獲得する魔法の手法はないのだから，そのような手法が紹介されると期待して参加した方々には申し訳なかった．しかし，日本において，いくつかの面白い成果が出始めている．そこで，それらを講演とパネルディスカッションをとおして紹介すること．それを「挑戦」として示すことがフォーラムの主旨であった．

1）ゴール指向分析，ミスユースケースについて

多発する要求変更へ対処することは要求工学が注目している大きな課題である．

古典的な要求工学では，開発の設計以降のフェーズから見た「何を作るのか（What）」を要求として定義することに主眼が置かれていた．ここ 10 年では，これを今一歩進めさせる手法として，ゴール指向分析に関する研究が数多く発表されている．設計が what に対して how を定義するプロセスであるとすると，ゴール指向分析では，why＝なぜシステムを作る必要があるのかを分析し，定義する．つまり，要求プロセスは，システムに対して存在理由や存在価値を与えるプロセスへと発展してきたと考えることができる．闇雲にシステムの要求を定義する前に，それが目的を達成するための妥当な解であるか否かを見極めるというのも，ゴール指向分析の役割である．Forum では，金庫問題[1]を例にして，要求工学の重要性を示した．開発したが使われないシステムというものは，このゴールが達成されないシステムであると考えることができる．システムを開発する目的が定義されれば，様々な代案を提示することも可能となり，それは開発後に多発する要求変更に対して，いくばくかの準備が可能になると期待できるのである．

また，システムとは，それが社会に導入されたときには誤った使い方や予期せぬ利用者，弊害などが様々な影響をシステムに与える．ユースケースが正常な使い方と，そのときに発生する異常事態への対処を定義するものであるのに対して，ミスユースケース[2]とは，望ましくない，あるいは誤ったシステムの脅威となる利用を定義したものである．ミスユースケースを定義することによって，その脅威を軽減させるための対処を新たな要求として定義することが可能となる．

これらは要求定義の新しい手段である．これらの手法の効果があるか否かをここで論じる必要はないと思う．その手法が目指すところが何れも，要求変更への対処や準備であることを考えると，導入すべき手法である．図 2 に，一連のゴール指向分析が貢献し得る要求品質を示してみた．この図は筆者によるものである．ゴール指向分析は，ゴールを設定

---

[1] 山崎利治: “DW2005 のための問題”, ソフトウェア技術者協会, SEAMAIL, Vol.14, No.15(01/2005), pp.51--52.

[2] Ian Alexander. Misuse Cases: Use Cases with Hostile Intent. IEEE Software, Vol. 20, No. 1, pp. 58-66, Jan./Feb. 2003.など

して，それを達成するための手段を分析するという枠組みを持つので，汎用性は非常に高いと言えよう．

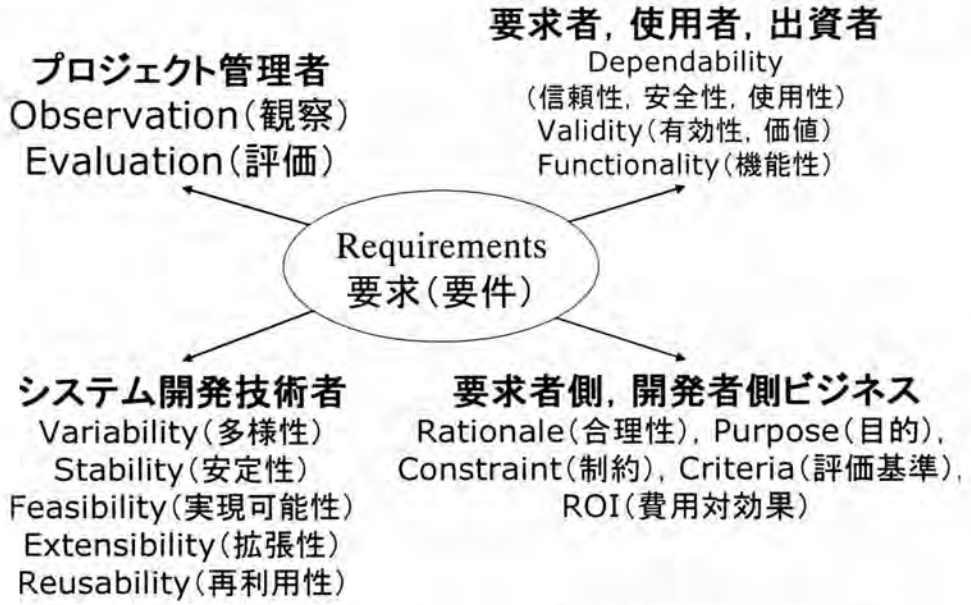


図 2　ゴール指向分析を適用すると，様々な要求品質向上が期待できる

2）超上流工程について

顧客は要求を発し，開発者は，顧客から要求を受け取ってシステムを開発する．そして，顧客はシステムを受け取って，「これじゃ使えない」と言うらしい．問題領域の専門家とはとても言えない顧客が世の中には沢山いるという話も聞く．一連の手法開発者の意図に反して，顧客はUMLを理解出来ないようだ．ユースケース（図ではなくて記述）を定義できない顧客が，"日本には"多いという諦めのため息も風の便りに聞いたことがある．まともなユースケースを記述できない開発者がいることも確かだが，それはここでは触れないことにしよう．とにかく，開発者から見た顧客は無知で，我が儘で，意地悪だと認識されているようなのである．

超上流工程とは，開発者が要求獲得のインタビューに来る前に，顧客企業側でやるべき事があるのだという警鐘である．開発の目的が定義されなくても開発者は，実は困らない．仕様に書かれたとおりのシステムを開発すれば，それで仕事を全うしたことになるからだそうだ．このような開発で実際に困るのは，顧客側である．より妥当なシステムの仕様があったはずであるのに，顧客側がシステムの目的を開発者に伝えないことが，不適切なシステムが作られる原因であるというのである．顧客教育という意味を考えると，要求工学が啓蒙すべき対象者は，開発者ではなくて，顧客企業側かも知れない．まさに，超上流工程は，要求工学が挑戦すべき分野である．

図 3 に，超上流工程が貢献できると思われる要求品質を示した．これは著者による分析結果である．黒字は貢献対象として期待できるものであるが，灰字は，恐らくあまり貢献

しないだろうというものである.

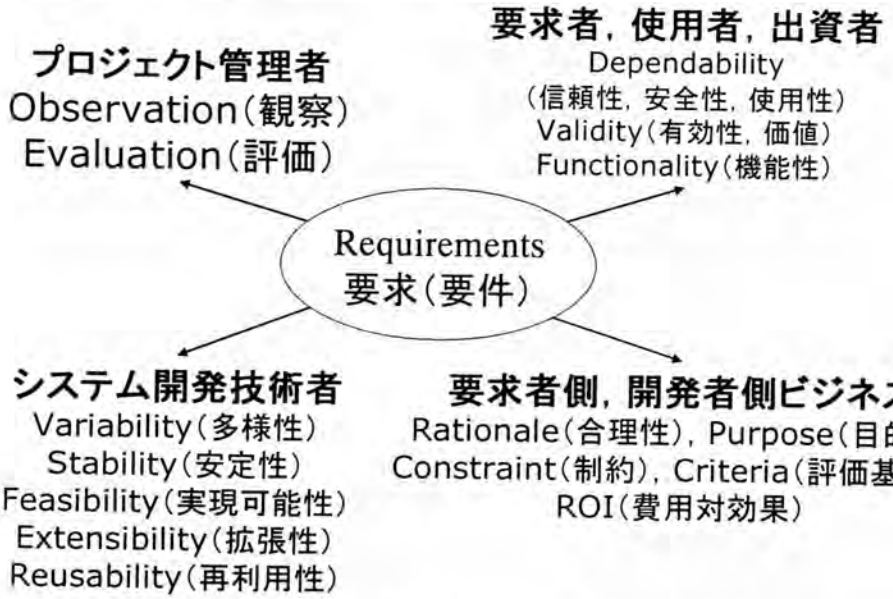


図 3　超上流工程が期待する要求品質向上に対する貢献

3）ジャステック法について

超上流工程は顧客教育という意味を持っているが，開発者には，上級の顧客と下級の顧客とを差別化する手段はないのであろうか.

ジャステックは，45 種類の顧客評価基準を定義し，開発コストの見積もりに顧客評価を反映させている．要求定義に際して，顧客側のキーマンが会議に出てこないといった声をよく耳にするが，それでは，そのような非協力的な顧客が，最終成果物たるソフトウェアやシステムの品質に与える負の影響を定量化するための努力を我々は行ってきたのだろうか.

要求プロセスは，顧客の協力がなくしては成功し得ない．それでも開発者にプロジェクトの成否の責任があるというのであれば，このような顧客の定量的な評価基準を設けることには大きな意義がある.

4）ビジネスモデリングについて

システムを開発するにあたり，そのシステムが導入される問題領域を理解することは重要である．また，実際に要求プロセスの会議に出席する顧客が，どのような立場の人々なのかを知ったり，顧客同士が相互の利害対立状況を理解したりすることは，システム開発を成功させるためには重要である.

ビジネスモデリングという用語は，一般に，新規ビジネスを打ち立てたり，ビジネスプロセスを改善したりするときにビジネスのゴールやプロセス，組織の役割を定義するために行われる作業を指す言葉である．転じてこの作業を開発者から見ると，導入するシステ

ムが，顧客側ビジネスへ与える影響を明らかにし，正当な要求を獲得しているか否かを検証するための準備や要求の根拠として行われるモデル化作業に相当する．

幸い我々開発者にはUMLという標準表記が手元にあるので，これを適用してビジネスの視覚化を行うことも可能である[3]．技術者は驚くかもしれないが，多くのビジネスモデルは，自然言語で定義されているのである．図表が用いられていたとしても，その表現力はお粗末な限りである．もちろん標準化などはされていないことが多い．ここに，モデル化を専門とする技術者が担うべき役割が見えてくる．要求獲得で使われる問題領域が，我々が理解できる（したくなる）形式で，正当に記述済みであったとしたら，その後の要求獲得と要求の仕様化のプロセスの生産性を，飛躍的に向上させることが可能となるであろう．図 4 に，斎藤氏によるビジネスモデリング手法の適用効果を示す．

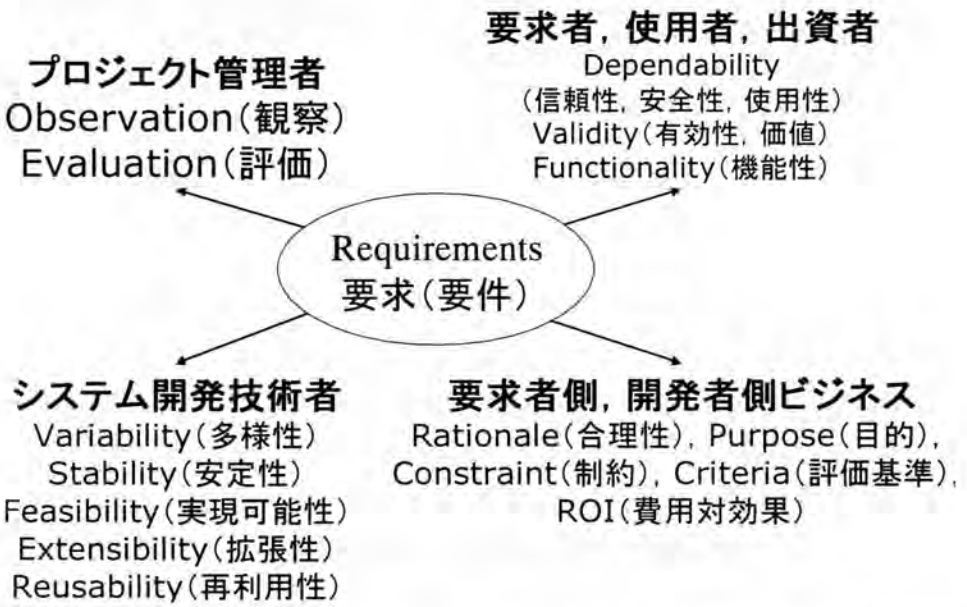


図 4 ビジネスモデリング手法 MOYA が期待する要求品質向上に対する貢献

5）非正常系の要求分析について

伝統的な家電製品となると，製品の機能や品質，性能といった要求は，設計へ至る過程で明確に定義されるようである．したがって，問題は，安全性を満たす設計を実現するための要求定義ということになる．これは主に，「××といった事故を起こさない製品」というように定義される．また，この要求は，設計を検証するために参照される．最近の家電製品の安全性を危うくしているのは，どうやら複数の（マイコンが組み込まれている部品も含めた）製品を統合したものが多くなっていることらしい．確かに，テレビは電波の受像器と表示器という役目を越えて，既にネットワークのインタフェースを備え，ハードデ

[3] Eriksson H. E. and Penker M.: Business Modeling with UML, John Wiley & Sons, 2000.(鞍田友美，本位田真一監訳: UMLによるビジネスモデリング，ソフトバンク，2002.)

ィスクを持つに至った．複合型製品では，故障や障害も複合化する．これは製品の信頼性を低下させることと同義であると想像するが，製造会社に対する世間の目は厳しい．製品の不具合はリコール対象となり，会社の存続をも危ぶまれるほどの損害が製造会社に科せられるのは，ご存知の通りである．この問題に対処するのも要求工学の一研究分野である．

複合型製品では，一つの構成製品のエラーが別の構成製品の障害の原因となり，最終的に製品の障害を起こすことも少なくない．設計では，それに対処するための方策を検討するが，要求段階では，設計された仕様をもとに，どのような障害が何を原因として発生させるかを探り，要求された安全性を満たさない場合は設計の改善を求めなければならない．このように，エラーから故障，障害へ至る連鎖を明らかにすることさえできれば，どこで連鎖を絶つかなど，必要に応じて適切な再設計が可能となる．

これには二つの方向がある．まず，システム内部で発生する現象と，それが現実世界に与える影響を「事象の意味」として対応づけること．これによって，エラー，故障，障害を解消すべきか否かの重要性を判断することが可能となる．もう一つの方向は，これらの現象が連鎖を明らかにするものである．いずれの方向も，実際の製品開発に導入され始めており，これまでの技術者依存の品質が改善されようとしている．

図 5 に三瀬氏による非正常系要求抽出で期待される効果を示す．

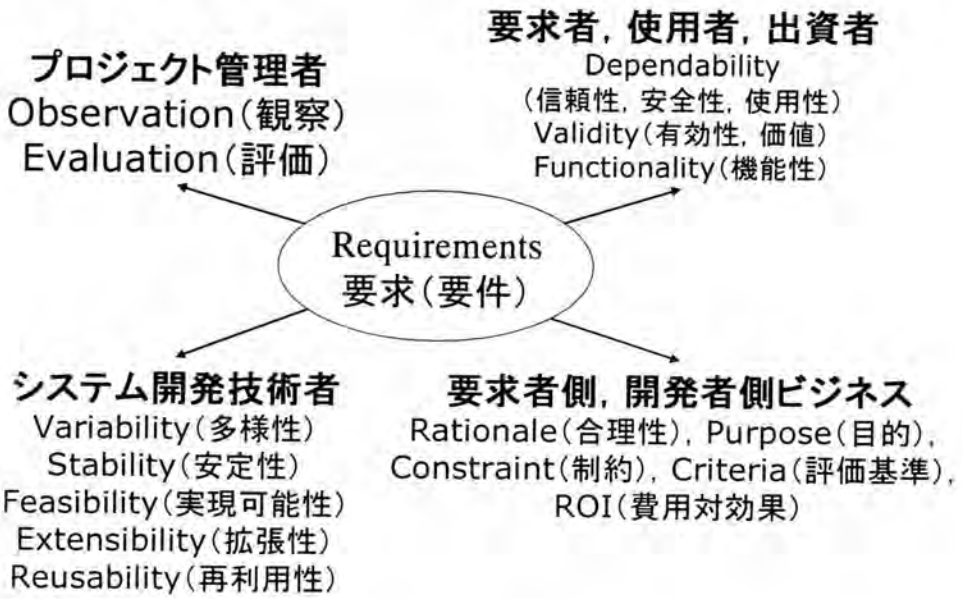


図 5　非正常系要求抽出が期待する要求品質向上に対する貢献

4．まとめ

フォーラムで御登壇願ったパネリストの方々は，要求工学が長年抱えてきた課題に対する様々な解や，解決の方向を示してくれた．これを実地で生かすのは我々の責任である．当日司会を引き受けた者としては，参加者によるアンケートを参照するにあたり，ここで

解説したような背景説明を当日行うべきであったと反省している．リベンジの意味も含めて，機会を作り，継続的な討論の場を設けていきたい．

謝辞

当日，ご講演いただいた信州大学の海谷治彦氏，SEC リサーチフェローであるアイネスの菊島靖弘氏，松下電工の三瀬敏朗氏，NTT データの斎藤信也氏，ジャステックの太田忠雄氏，そして，質疑にご参加いただいた会場の方々に感謝いたします．

「SEA Forum August これからのシステム開発:要求工学の挑戦」始末

藤野 晃延
有限会社インアルカディア
pvalias_at_acm.org †

1. はじめに

熊本にて今年6月に開催されたソフトウェアシンポジウム、そのワークショップ Group-H のテーマは要求工学であったが、参加者の一人のポジションには、「誰も必要としていない物を正しく作る空しさからの脱却」と記されていた。折角、作るのであるなら、確かに人から必要とされる、そしてこの世界の何らかの「役に立つ」システムをこそ、是非、作りたいものである。

しかしそのようなシステムを作るのは、実際には必ずしも簡単ではないと見える。例えば以下はアフガンにおけるタリバン掃討作戦中に起きた米軍の「友軍誤爆」に関するワシントン・ポスト紙のコラム記事[6]なのだが。

空爆すべく敵の位置を高精度携帯 GPS 受信機—通称:プラガ—に入力した送信係がその「発射」ボタンを押すと、バッテリ低下を示す警告ランプが点滅した。送信係は予備のバッテリに交換した後、再び「発射」ボタンを押した処、やがて友軍からの2,000ポンドの空爆が彼らを襲い、送信係を含む特殊部隊員 3 名が命を落とし、20名が負傷した。プラガは、バッテリ交換の後では、機器自体が今在る位置へと座標情報が初期化される仕様となっていたが、そのことに送信係が気付かなかったことが原因であった。極度の緊張が強いられる戦場ではこのような「軽微な過ち」は起こりえるものであり、今回の件を軍の上層部は特に規律違反ないし破廉恥行為などとは見ておらず、それよりはプラガを取り扱う兵士たちへの教育を徹底する必要性に言及している。

さて、皆さんはどう思われたでしょう。少なくともこの当の連絡係が「必要」としていたのは、自分を誤爆するようなシステムでなかったことだけは確か。哀号。

要求工学 (以下 RE と略す) は、この必要とされ、そしてこの世界をより良くするシステム、あるいはソフトウェアの実現を望む人々に、進むべき方向と、頼りになる道具を必ずや提供してくれる筈である。

2. 要求工学とは

フォーラムの報告に入る前に、要求工学に関してその定義、幾つかの基本的な用語と概念、および範疇を解説したBasharらの論文[1]から、要点を掻い摘んで記す。これはミレニアムを記念した ICSE 2000 において展望を含めた RE ロードマップとして提示された論文である。

2.1. 背景

ソフトウェアシステムの成功の尺度は、それが意図された目的にどの程度、合致しているかに掛かっている。端的に云って RE とはこの目的を発見するプロセスのことであり、即ちステークホルダとその望んでいる事柄を明らかにし、分析、意思疎通、および後続する工程がそれを核として進めることができるような文書をものすことである。

しかしこれには数多くの困難さが伴う。ステークホルダは多肢に渡り、またあちこちに離散していることもあろう。彼らがゴールとする処は、彼らが働き、また遂行することを欲している業務を取り巻く環境を、彼ら自身がどのように捉えているかに拠り、幾つかの異なったものとなろうし、あるいは互いに相容れないものとなることもあろう。多視点的アプローチが本質的に必要とされる所以である。

2.2. 要求工学の定義

古典的な Zave の論文に因れば RE の定義は、

「RE は、ソフトウェアシステムに就いて、それがゴールとしている処のものは何か、どのような機能を有するのか、そしてどのような制約が掛かるのかを検討する、ソフトウェ

† Spammer 対策のため、実際のアドレスは at を@に置換願う。

ア工学の一部門である。加えて、ソフトウェアの振る舞いに就いての仕様の精度と、時間経過やソフトウェアファミリに跨る発展に対し、前述の要素がどのように関係するのか、を同様に検討対象としている。」

となっている。

この定義の魅力は幾つかある。先ず「現実世界のゴール」を明らかにすることの重要性に言及していることであり、これが動機となって即ち意図されるソフトウェアシステムが構築されることになる。これらゴールにより「何を(what)」と同時に「なぜ (why)」が示されることになる。

二番目に「仕様の精度」への言及がなされており、即ちこれにより、要求の分析と、それが紛う方なくステークホルダが望んだ事柄であることの実証と、何を設計者は構築すべきかの定義と、そしてシステムの提供時に開発者らが正しく為すべきことを為したことの証明、これらの拠って立つべき基盤が提供される。

三番目に、時間の経過やソフトウェアファミリに跨る発展への言及により、時々刻々と変化し続ける世界の真の姿と、他の工学分野の技術者が往々にして行っている部分的な仕様の再利用の必要性が力説されている点が素晴らしい。

2.3. 要求工学の工程

中核となる RE 活動は、

- 要求の引き出し
- 要求のモデル化と分析
- 要求の伝達・意思疎通
- 要求の合意
- 要求の発展

であるが、実際には、これらは相互に絡み合っており、イタラティブであり、そしてソフトウェア開発の全ライフサイクルに跨ることもある。

2.4. 要求工学と開発との関係

典型的な工学の定義は、科学的知見を用い、実践的な課題に対する経済的有効性を有した解法の構築を行う行為を指すことから、 RE を工学と呼ぶのは妥当でないとの非難もあるが、 RE は工学的工程の重要な部分を担いでいると同時に、開発活動を実世界の問題と深く結びつける碇の役をも果たしている、と云う特質を忘れないための「証左」として、工学と云う用語が RE に付けられているのだと捉えることができよう。現実の課題に根ざしているからこそ、解法の妥当性と経済的有効性が分析できると云うものだ。

同様に RE に工学と云う用語が付随する理由として、仕様それ自体は工学的に追求されるべきであり、それ故 RE は解かれるべき課題に就いての認識から、その課題に就いての詳細な仕様に至るまでの、これら一連の工学的意思決定を表現している、との見解が関係している。

2.5. 多元的学際領域としての要求工学

RE は多元的学際領域であり、また人間中心の工程でもあり、そこで用いられる手段や技法は様々な学際領域から持ち込まれていることもあって、RE 技術者はこれら様々な分野に通じた技量を有していることが期待されることとなる。

RE はその文脈として、ソフトウェア開発、システム工学、そして人間活動システムとから接近することができる。RE がその役目を果たす文脈は、通常、人間活動システムであり、それは課題の所有者は人だからである。

また、ある一つの RE 活動に従事する際における予見として、どのようなコンピュータベースのシステムであれ有意義足り得るが、システムの導入により、それが支援の対象としている活動自体を変貌させてしまう。それ故、RE は、人がどのように彼らを取り巻く世界を感知し、理解するのか、彼らがどのように相互作用するのか、そして職場の社会学が彼らの活動にどう影響するのか、これらの事柄に対して繊細な配慮を行う必要がある。

さらに、要求の引き出しとモデル化に関しては、その理論的根拠と実践的技法の提供に当たって、 認知科学および社会学が RE で用いられる。

認知心理学 (Cognitive psychology) : 人がその要求を示すことに困難を憶えることがあることに関しての理解を得られる。例えば問題領域の専門家はしばしば、自覚が困難であるような多くの暗黙知を有していることがあり、そのような場合には RE 技術者から提示された質問に対する回答が、彼らの実際の行動とは合致しなかったりする。

人類学 (Anthropology) : コンピュータシステムがどのように人の活動を支援、あるいは妨げるか、に就いての深い理解を得る一助として、人間活動観察のための方法論的接近が可能となる。

社会学 **(Sociology)** : コンピュータ化により励き起こされる政治的あるいは文化的な変化に就いての理解が得られる。新しいコンピュータシステムの導入は、その組織において実施されてきた業務の本質を変えてしまい、組織構造や意思疎通経路に影響を与えることもあるし、当のシステムがそれを満たすために造られた筈の当初の要求さえもが変貌してしまうことさえある。要求収集の実行は従って政治色をも帯びることがある。

対処法としては、成り行きから最も大きな影響を蒙る当事者を要求定義工程に参画させることを意図した、所謂「スカンジナビア的 (Scandinavian)」接近法などが含まれる。

言語学 **(Linguistics)** : RE の大方は意思疎通に関わることから、言語学は重要である。言語学的分析に基づくことにより、例えば仕様の表現における言語の用い方が変わり、曖昧性が排除され、理解容易性が向上されたりする。また組織における意思疎通パターンの分析などの言語学的手法も、要求引き出しにおいて活用できる。

最後に、RE における重要な哲学的要素に触れる。RE はステークホルダの用語、概念、視点、そしてゴールを、解釈し、そして理解することに根差した活動である。即ち RE それ自体が、ステークホルダが有している信念 (Epistemology: 認識論) への理解、実世界における観察可能性とは何か(何であれば観察可能なのか)と云う問い (phenomenology: 現象学)、何であれば客観的真実として合意可能 (Ontology: 存在論) なのかと云う問い。これらと真摯に向き合うことが不可欠である。これらは要求を検証しようとする際、常に直面することになる重要な課題である。特にステークホルダがそれぞれ異なるゴールと共訳可能でない信念システムを有している場合には殊の外、重要となる。

尚、これらはモデル化技法の選択においても重要である。ある技法の選択により、モデル化できる対象としての現象の集合に影響が生じ、要求分析者の観察能力を制約することさえ起こり得るからである。

フォーラム: 要求工学の挑戦

海谷氏の発表

【発表の論旨】: セキュリティと云う分野を前提とした際の、要求表現手段としての既存ユースケースを拡張した表現手段の可能性を述べる。

【背景】: セキュリティ分野での効果的な要求表現としてユースケースに対する拡張が提案されてきている。

1. Abuse case (※ /abju:z/ ではなくアブ・ユースケースと呼ぶ)
2. Misuse case (ミスユースケース)
3. セキュリティ・ユースケース

【動機】: 以下のセキュリティ要求 3 種、これらを明確に表現したい。

Confidentiality:	秘密を暴露するな
Availability:	邪魔するな
Integrity:	壊すな

【フォース】: (セキュリティ専門家でない)ステークホルダにもわかり易い説明・提示法が必要

【解法】: ユースケースを利用した獲得・分析法が有効

【例示】: ミスユースケースの例

- 害となるユースケース, 害をなすアクターを構文的に分けて書く
- 通常なものと害なものを 1 つの図に書く
- ある Misuse Case がユースケースに脅威を与えることを threaten という関連で仕様化
- あるユースケースが Misuse Case の脅威軽減となることを mitigate という関連で仕様化
  - コレはセキュリティケース(後述)に相当する

図 1 ミスユースケース記述例-1

安全要求についても対応可能

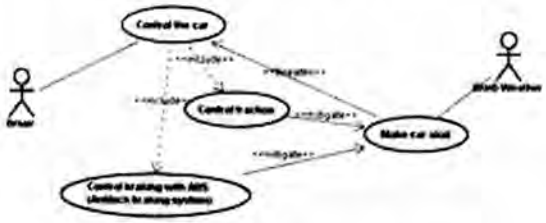


図 2 ミスユースケース記述例-2

【筆者感想】: abuse case は悪意の有る、ないし好ましくない利用の仕方、を行うアクタを明示的に切り出し、そのアクタの励起できるユースケースを記述することによりセキュリティに対する脅威を明確に仕様として列挙しよう、と云う戦略と読めた。

問題を明示しようとする姿勢は評価できると思うが、しかし個人的にはそれら脅威が何ゆえ、脅威となるのか、「悪いことは悪い」と云う常識論と云うかセキュリティ分野の知見なのかも知れないが、そのような「金科玉条」的な視点からではなく、本来のステークホルダが達成を欲しているゴールに対し、どう関係し、どう影響し、従って確かに「脅威」となるな、と云う意味的に納得できる連鎖と云うか合理的関連付けが欲しい、と感じた。

これに比べるとミスユースケースの方は、アブユースケースに欠けていたゴールとの関係性と云う面が表現できるようになっているため、普段、要求やゴールの存在範囲に就いて、望まれるゴール／望ましくない避けるべきゴールと云った両面から認知地図もどきを先ず求める、としている筆者の接近法と内容的に類似しているせいか、直感的にも「ふむ、こちらはそれなりに効果的なのでは」と云う好印象を憶えた。

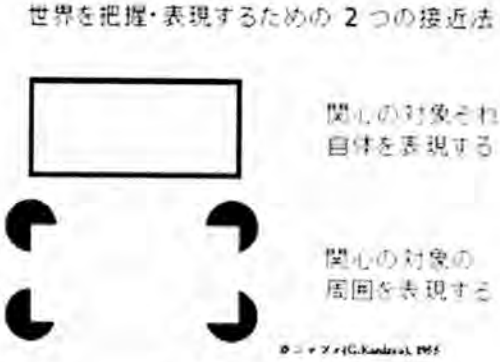


図 3 二種類の表現

世界を表現するには大別して 2 つの接近の仕方があると筆者は考えており、それはゲシュタルト的になるが、関心の対象を直接、指示する方法と、対象そのものの代わりにそれを取り囲む周辺を示すことにより、対象を「浮かび上がらせる」、と云う方法である。模式的にその相違を示す。これは「カニッツアの錯視図形」として有名なものの一つであるが、周囲の 4 つのパックマンを描くことにより、「存在しない筈の四角形」が我々には明らかに「見え」る。

通常のユースとミスユースの両者を求めるミスユースケースも同様の働きを、実用としての側面のみならず、恐らくステークホルダに対し、対象とそれを取り囲むモノ(一種の文脈とみなせる)、と云うペアの情報で把握できると云う、認知的側面からの利点をも提供しているのではないだろうか。

今回はセキュリティ分野を適用の範疇として絞り込んだ適用例であったが、ミスユースケースに関しては、セキュリティ分野に留まらず一般的な要求の獲得や検討の際にも適用が有効であるように感じられる。今後、一般のユースケースに基づく要求工程の作業と共に用いたケーススタディが行われることが望まれる。

3.2. 菊池氏の発表

【発表の論旨】: 超上流から攻める IT 化の勘所
【背景】: 既に IT 化がそれなりに浸透し、IT 抜きでは一歩もビジネスが進まない組織、(保険・金融サービス?) IT 的重厚長大産業、業務としての成熟度はそれなり。
【文脈】: IT のモデル化は行っているが、ビジネスのモデル化は行ってきていなかった。
【フォース】: システムが外に飛び出し既に社会基盤化しており、それなしでは何も進まない、利用者として市場が直接、システムに触るようになり、多様化した要求に応えねばならない。
【問題】: 造り始める前に勝負が決まってしまう、超上流工程プロセスが開発品質を決めてしまう、
要件定義が困難になってきている
システムを作るまでは可能性の塊だが、できあがると制約と条件の塊
作るほど有能な人を拘束する
【解法】: 要件定義がシステム開発力強化の鍵、
要件定義力をつけ、発注力を強化しよう、
企画発注部門の IT 力強化、超上流工程の強化
IT 部門の強化

【筆者感想】: 「超上流工程」と銘打った菊島氏の講演であった。その主張には大いに同意できる部分が幾つかあったが、それ故に尚更、全体としては「成熟産業」、別な言葉で云えば「IT 重厚長大産業」、における IT 化のある種、本質的な普遍性のようなものを垣間見せられた思いがし、一昔前、ダム建設など土木工学の巨大プロジェクトで膾炙していた「鉄則」が脳裏に蘇った。土木工学の巨大プロジェクトの成否は、将に実際の建設作業が始まる前に、その準備が整えられるかどうか、に掛かっていた。例えばダムの場合、現場までの建設用道路確保、必要な建設機材、材料をダム建設現場に搬入・準備するまでに、概ね計画の 2/3 の時間が費やされると云われており、つまり実際のダム建設の時間は全工期の 1/3 でしかないのだ。

IT 化への投資がそのままビジネス上の優位性に結び着く分野では、確かにトップの IT 化に関する決断が全てを決めると云え、その意味で所謂 90 年代初頭までの MIS 的な世界観と構造がそのまま継承できているともみなせよう。ここでは最も主要で決定的な決断を行うステークホルダとして組織トップが頂点に存在している。この点が昨今の RE で克服すべき課題として重要視されている多様なステークホルダが絡む、多視点とそこから発生する衝突可能性の事前検出や衝突回避、あるいはネゴシエーションと云った問題が、決定的なリスクとして浮上することなく解消できている背景をなしているのであろう。

筆者としては、巨大ビジネス系システムに共通する「普遍性」が、折角の要件定義の強化も、結局、本質的にモノリスティックで巨大で複雑なシステムの維持拡張の手段としてしか位置付けられていないのでは、との印象を受けるのだが、思い違いであろうか。

3.3. 三瀬氏の発表

【発表の論旨】: 非正常系シナリオ: 民生商品での非正常事象の要求仕様抽出
【スコープ】: 民生品を対象とした組込みシステム
【文脈】: 不特定多数の人により不特定多数の環境下で使用される
⇒ 組込みシステムでは、徹底した安全性が要求される、多少故障しても何とか使い続ける(寿命末期の安全性)、生活の基盤となっているものなど簡単に機能停止できない
間違った使い方、間違った目的で使う(イタズラを含む)、極めて僅かでも安全上の問題を起こせない
Cf. 石油ファンヒータ、ガス湯沸かし器、エレベータ・・・
【フォース】: 組込みシステムの安全性とその他の品質トレードオフでは許されない、両立が要求される
- 出来る限り動作させる
- 耐久性が高い
- 不便にならない(使用性)

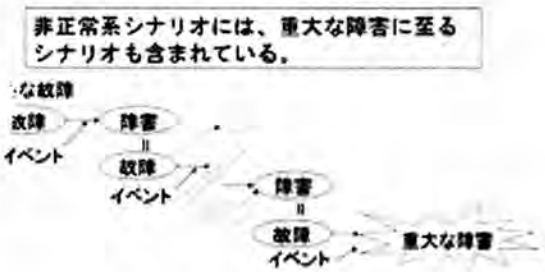


図 4 非正常系シナリオ

【問題】: 予測していない状況が存在すること
ソフトウェアは、仕様の通りに動作する。仕様に記述されない状況が発生した場合、どうなるか分からない。
【解法】: 障害を予測さえできれば、それへの対策はシステム毎にどのようにでも対策が立てられる。
但し、出来る限り早い段階で予測しなければならない
その対策がハードウェアの設計変更を含むことになると、問題は更に大きく、複雑になる。
【例】: 障害に至るシナリオとその重大性の認識がないと、原因の対策に抜けが発生する

あるシステムは、開始ボタンと終了ボタンで操作される。誤操作回避のために、2つのボタンの同時押しを受け付けない仕様になっている。
障害発生!開始ボタンを押したら、ボタンが戻らなくなった。(その結果)システムの動作を止めることが出来なくなった。火を消せない、ガスを止められない、機械を止められない。
重大性の認識→対策:停止ボタンは、必ず実行される仕様へ。

【問題2】: 重大な障害シナリオを定義する技術は設計者の経験に依存しており、障害シナリオを定義する手順も定義されていない。
【フォース2】: 対処すべき非正常系現象は、設計者により予測結果のバラツキが大きい。
レビューを十分に行えない。

ノウハウが、組織的に蓄積されにくい。

【参考】: 設計段階の障害分析法
FTA (Fault Tree Analysis)
FMEA (Failure Modes and Effect Analysis)
HAZOP (Hazard and Operability Analysis)

【解法2】: 非正常分析手法例(ESIM)
非正常現象の抽出
1. システムの各構成要素で発生すると予想される非正常現象を抽出
2. 構成要素間で発生すると予想される非正常現象を抽出、非正常現象と、それへの対処方法、その重要度を分析
3. 対処できていない非正常現象がシステム全体の各構成要素へ与える影響を分析
→新しい(=当初は予想できなかった)非正常現象を抽出

2～3を繰り返す→重大な障害を発生させるシナリオを得る
どこで非正常現象の連鎖を止めるかを検討する。
ハードウェア再設計、またはソフトウェア設計で障害シナリオへの対処方法を取り込む

【問題意識】: ユビキタス社会に進展していく中で安全が確保できるか。
もちろん問題は、安全をベースにした全ての品質
現在想像していないものが、同一ネットワークで接続
社会インフラ(社会基盤として、停止できない、安全確保)
オフショア開発が進む中で、誰がどのように品質を保証、IEC61508(機能安全)の動向

【結論】: 情報の高度化と社会インフラ化が進展し、従来の開発業務全てが、情報処理技術の視点から抑えないといけない。
従来の情報処理技術者の役割範囲を超える必要があり、要求工学の果たすべき役割が最も大きい。

【筆者感想】: 三瀬氏の発表において言及されているような所謂、生活環境に埋め込まれ、しかも同様に新たに生活環境に埋め込まれてくる他のシステムとも有機的なネットワークを構成しなければならず、加えてそのような生活環境自体の「発展」により生じる事象の意味的側面を破綻することなくケアしなければならない、と云う本質的に全体の俯瞰や前以ての予想が意味を成さないシステム環境における RE 活動は、将に RE 最前線、と云う趣が漂う。

しかし今後の難問は「意味」の部分で、これは実世界の system (生物や自然) level では「曖昧なまま」に通用している、と云うか自然は実によくできていて、それで通用するようになっているのだが、しかし digital world は今のままでは全てを明確に切り分けなければならず、その切り分けは常に「一意」であることが前提となってしまっている。

例えば「同じ」とは何をもって同じとするのか、1 meter は本当に厳密に 1 m でなければダメ (誰もが原始時計や水準器を持ち歩かなければならなくなる！) なのか、普段、我々はそれほどの「峻別」は必要としていないのだが。

3.4. 斉藤氏の発表

【発表の論旨】: 要求定義の質的向上を狙うビジネスモデリング方法論 MOYA

【文脈】: SIer として顧客要求に応えたい

【動機】: どうすれば仕様は安定するの？
頑張って仕様を精緻且つ大量に書く (IEEE 830-1998, Volere Template, etc.)、しかし却って要求は不安定に。
仕様を書くのはあきらめ、作ってみて仕様を確認する
(費用、アーキテクチャに無理が)

【問題】: 少ない記述量で十分に理解しあうためには？
明示的に書いていないことを理解する

【解法】: 要求仕様の「背景」を理解することで記述の不足を補う、「業務を知る」「お客を知る」

【問題2】: 何を知れば「仕様の背景」を十分に知ったことになるの？
仕様の背景はどう表現・モデル化し、お客と確認しあえばよいの？

【スコープ】: 「仕様の背景」とはお客様の課題、システムの目的、業務フロー辺りが核

【解法2】: 背景を理解するために行うべきこととモデル化の対象、ビジネスモデリング方法論 MOYA (Model-Oriented Methodology for Your Awareness)

関係者、課題、目的、手段、業務をモデル表現し、吟味する手順を定めることで、「気付き」を促し、要求定義の質的向上を実現する

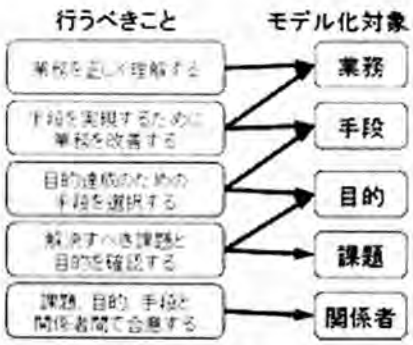


図 5 モデル化対象と行うべきこと

【由来】: 様々な手法の統合:
ソフトシステムズ方法論 (SSM)
ゴール思考分析、
エッセンシャルユースケース、
UML 多視点モデリング

【展望】: 現在までに 20 ほどの適用事例、その結果を MOYA ナレッジベースとしてイントラ公開、ノウハウ共有を可能に。

【筆者感想】: Hoare の言葉だったと思うが、「設計には二つの種類が存在し、一方は十分簡明なので欠陥が存在していないことが明らかであり、他方は、余りに複雑なので明らかに判る欠陥など全くない」とのことだから、「精緻に大量に書く」とのアプローチは後者の種類に分類される典型だろうから、将によい時期に適切な決断をされた、と云えるのではないでしょうか。

3.5. 太田氏の発表

【発表の論旨】: 要件を定量的に捉えた見積りモデル: ジャステック法
(要件:品質要件および開発環境要件)

【文脈】: 要求定義が済み、要求仕様が完成している段階での見積もり。

【特徴】: ソフトウェア開発を生産工程管理と捉え、開発計画とその見積もりに準じたプロジェクトの開発計画達成のためのコントロール(フィードバック)が可能となる、フィードバックは当該プロジェクトか、あるいは以降の同類の開発プロジェクトに懸かることになる。

【筆者感想】: 要求定義が行なわれた後、の見積りと管理の話題(RE前、RE後の様相の分水嶺を意識する事は大事)、実際に見積り情報が蓄積されれば、定量化した見積もりは、一種の「平常値」として機能する可能性があり、その意味で変動に気付く良いインジケータであると云えよう。

3.6. パネル討論: Q&A

パネル討論自体は、残念ながら余り活発な議論に至らずに時間切れが来た、と云う状況だったように思う。議論にならないのは当然で、要求工学に関しては恐らく、誰も「反対」意見を持つことはないだろう。やらなければならない事は山のように残っているにしても、REに関わるあらゆる取り組みは、行うべき価値があるのだから。

1 つ、会場から、要求仕様書が書けない技術者に、どう指導したら書けるようになるだろうか、と云う質問が為された。

残念ながら会場のその場では、明確な有効策は応えられなかったように憶えているが、それでもユースケースの記述など、テンプレートに準じて記述する訓練は、それなりに効果があるだろう。しかし要求仕様を、必要十分な内容を備えた仕様として残せるようになるには、それなりの経験が必要であろうと筆者は感じている。

冒頭に掲げたGPSプラガの友軍誤爆だが、バッテリにデフォルト値を持だせること、バッテリ交換後は、そのデフォルト値が有効となること、デフォルト値は、そのプラガが在るその位置とすること、など、これを要求として明記することは難しくないだろう。

しかし仕様を考えた者は、戦場で「発射」ボタンを押下すべき状況下にある兵士が、バッテリ交換の後にどのような行動を最も自然に採るか、と云うことに思いが回らなかったのだろう。つい今しがた、自身が入力し設定した座標情報なのだから、バッテリを交換したとは云え、僅かの時間しか経過していないのに、その情報が綺麗に別の情報に書き変わっているだなぞ、一体、誰がそれを「自然」に考え付くと云うのだろう。

しかしプラガを笑うことはできない。今のインターネット上のサイトでも、折角、利用者が入力した情報を、一箇所の軽微な入力間違いだけで、残る正しい情報を全て綺麗に初期値であるブランクにして「入力間違いがありました、読みはひらがなで入力してください」などとのメッセージを詫びるでもなく平気で表示しているサイトもあるのだから。

このような機能的には本質的ではないが、しかし認知的に自然でないとか、あるいは利用者にストレスを感じさ

せたりしてしまうような設計を仕様から排除するには、三瀬氏が発表に言及しているように、非正常系のシナリオを綿密に検討するのが最も効果的であると、筆者も思う。その意味で要求仕様を担当する人は、シナリオのバリエーションを思い付ける想像力のある人こそ適性があるのかも知れない。

4. 付録: 要求工学に関する個人的感想

4.1. 開発側からばかり見ているような気が。。。

20 年ほど前, 京都の会合で当時シュルンベルジュ・ドール研究所の D. Barstow が次のような模式図を示しながら, ソフトウェア工学の進捗とゴールに就いて語ったのを憶えているが, 今の要求工学にもこの図式がほぼそのまま当て嵌まるように思える.

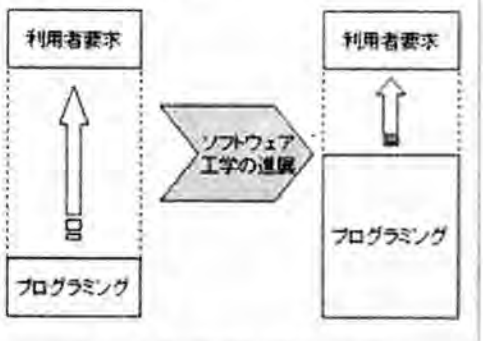


図 6 要求と開発の溝:模式図

他にも例えば概念プログラミング[2]など, 当時は専らソフトウェア工学の技術側面から, 利用者要求にどう接近するかが, 主要な関心事とされていたように記憶している. 利用者側の要求や期待に対し, 常にシステムないしソフトウェアを構築する側からの視点からのみ, 接近してきたと云え, それは例えば上記図中にもその傾向はハッキリと窺い知れる.

80 年代末, オブジェクト指向による大規模なクラスライブラリの開発／管理／適用に従事していた筆者は, B. Meyer[4]の「クラスの継承とはクラスを開発する者のための機構であり, クラスを利用しようとする者のための機構ではない」との言に接して愕然としたが, この指摘は今のコンポーネント技術の必要性と出現を予言したものと云え, インタフェースと云うクラス継承とは別の機構が利用する側に提供されるようになった.

同様に要求工学においてもこれまでの「作る側の視点」中心的な接近を捉え直すことにより, 進展が望めるのではないかと筆者は考えており, 特に次に示す大きな難問の解決には人類学や言語学と云った学際領域の知識を総動員した接近が不可欠と云えよう.

4.2. 二種類の境界の設定と云う難問

要求の一つの側面を把握する様式としてユースケースの有効性が知られている. ユースケースはシステムの利用者と利用されるシステムとの境界であるシステム境界(下図の①)を明らかにするのと同義と見なせる. またこのシステム境界は, M. Jackson が問題フレーム[3]においてマシン境界と呼ぶものとも同義と見なして良いだろう.

図 7 二種類の境界

システムにしろマシンにしろ, それらは実世界の裡に措かれ利用される分けだが, ではそのシステムないしマシンからの影響を受ける実世界の方は何処までを勘案して考慮すれば良いのだろうか. 図中②の境界を定めないままでは, 適正な①の同定は覚束ないであろう.

筆者の乏しい個人的体験からは, この外側の境界が合理的な根拠の許に定められた開発事例はなかった.

Zave[5]は要求工学をどう分類するか, と云う視点からの先駆的論文で, 恐らく初めて当時の要求工学のほぼ全範疇を捉えようと試みたが, しかし後にミレニアムを記念した ICSE においてソフトウェア工学の分野毎に示されたロードマップの一環として要求工学に関し Bashar らにより示されたロードマップ[1]は, 明らかに Zave らにより示された範疇より拡大されていた.

同様にその暫く後に I. Sommerville により示された要求工学に就いてのロードマップは, 明らかに Bashar らのそれより地平は広がっていると筆者には読めた. これらロードマップないし定義や分類に関する論文の言及する要求工学の範疇がこのように拡大／膨張を続けているように見える背景には, 先に述べた二種類の境界の裡の外側, その境界を捉えることの困難さと通じているように筆者には映っている.

参考文献

[1] Nuseibeh, B. et al. Requirements Engineering: *A Roadmap*, ICSE, 2000

[2] Hartley, R.T., *An Overview of Conceptual Programming*, Conference on Intelligent Systems and Machines. Oakland University, Rochester, Michigan, April 1986

[3] Jackson, M., *Problem Frames*, ACM Press, 2000

[4] Meyer, B., *Object-oriented Software Construction*, Prentice-Hall, 2000

[5] Zave, P., *Classification of Research Efforts in Requirements Engineering*, ACM Computing Surveys 29 (4), 1997

[6] Loeb, V., *'Friendly Fire' Deaths Traced to Dead Battery: Taliban Targeted, but U.S. Forces Killed*, Washington Post, Sunday, March 24, 2002; Page A21
http://www.washingtonpost.com/ac2/wp-dyn/A8853-2002Mar23?language=printer

SEA Forum Aug.
要求工学

海谷治彦氏(信州大)
菊島靖弘氏((株)アイネス)
三瀬敏朗氏(松下電工(株))
斉藤信也氏(NTTデータ(株))
太田忠雄氏((株)ジャステック)
(中谷多哉子(筑波大, エス・ラグーン))

Group Hまとめ:認知マップ

ワークショップからの提言

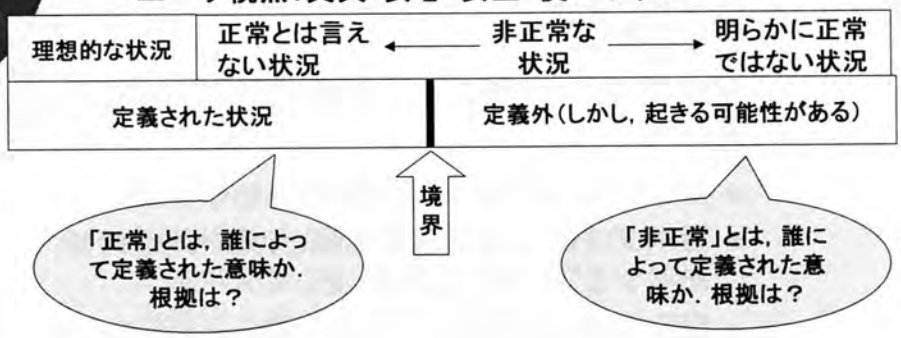
- 誰も必要としていない物を正しく作る空しさからの脱却
- 要求工学が対象とする世界は広く，問題の根は深い
  - 要求の起源を求めると際限がないが，
  - 要求プロセスとは，現実世界との関わりを吟味，熟考するプロセスである．そして，
  - 現実世界の意味をシステムに取り込むプロセスとしての責務は重大である．

本Forumの目的

- 要求工学の現状と課題，適用可能な最新技術の紹介
- 技術の有効性を「要求」の意味から評価するための討論を行うこと
- 「意味（主観）」と「仕様（客観）」の境界をどう取り扱うべきか．越えられるのか．越えるべきか．
  - 個々人が思いこんでいる「現実世界の意味」を取り込むことなどできるのか？
- 開発への取り組みは？

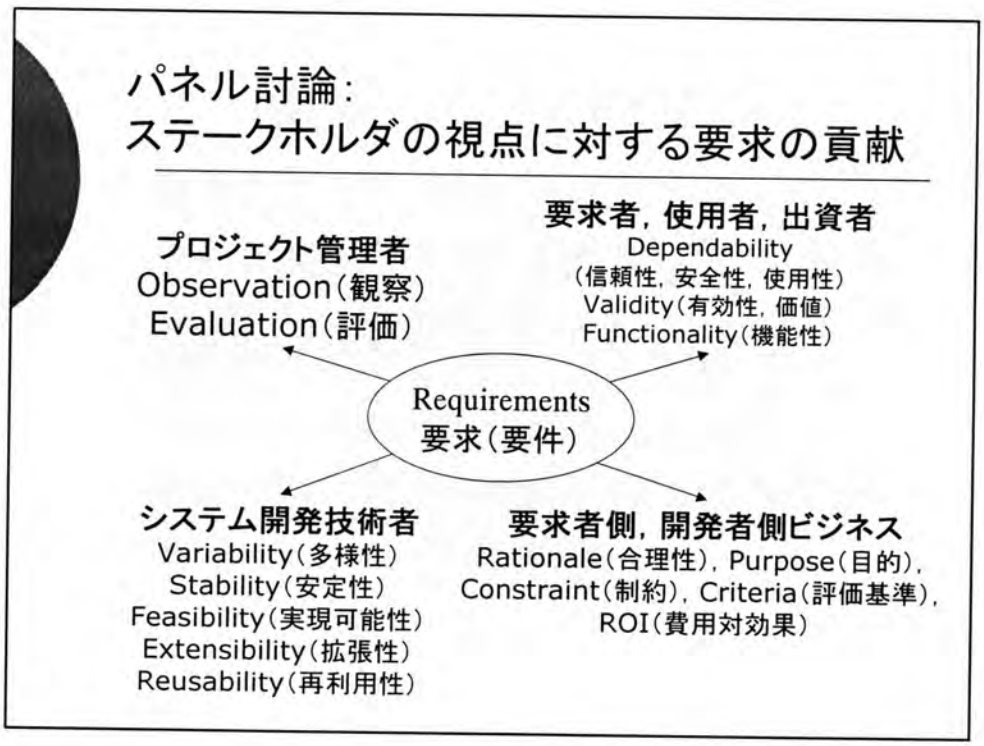
現実世界の意味とは. . . 例

ユーザ視点:丈夫・安心・安全・使いやすい



用語の確認

- 要求:requirement
  - 要求者←→分析者
- 要求, 要件:requirements
  - 分析者←→設計者


パネル討論:
ステークホルダの視点に対する要求の貢献
プロジェクト管理者
Observation(観察)
Evaluation(評価)
要求者, 使用者, 出資者
Dependability
(信頼性, 安全性, 使用性)
Validity(有効性, 価値)
Functionality(機能性)
Requirements
要求(要件)
システム開発技術者
Variability(多様性)
Stability(安定性)
Feasibility(実現可能性)
Extensibility(拡張性)
Reusability(再利用性)
要求者側, 開発者側ビジネス
Rationale(合理性), Purpose(目的),
Constraint(制約), Criteria(評価基準),
ROI(費用対効果)

ミスユースケース
セキュリティ要求および他

21 August 2006

海谷 治彦

1

講演者の概要

- 地方大学の先生です.
- 先生は12年くらいやってます.
- ソフトウェア工学を専門としています.
- 割と昔(博士の学生の頃から)要求工学をやっています.
- 詳細は以下を参照ください.
  http://www.cs.shinshu-u.ac.jp/~kaiya/
  ただし8月中は事情により上記は休止中.

2

追求している問題点

- 開発における妥協や合意の達成.
- 既存資産(知識)の有効利用に基づく開発の効率化, 競争力強化.
- 技術の段階的な適用による浸透.
- 技術利用者(技術者)個々人の納得.
- 要求定義段階でのエフォートの効果を調査.

3

研究テーマ群

- 要求工学
  - ゴール指向分析
  - オントロジーに基づく要求獲得支援
  - 既存システムの特性に基づく要求定義支援
  - 要求工学品質と製品品質の関連調査
- ソフトウェアプロセス
  - PSP(個人ソフトウェアプロセス)を用いた手法評価.
  - 開発文書の段階的な変更に基づくプロセス改善.
- ソフトウェアセキュリティ
  - 要求・脅威に基づくポリシー定義

4

熊本グループHについて

- 参加できなくてすいません.
- 横軸について
  - 社会学的部分は重要かつ(学者としては)興味深いですが,
  - 工学的(系統的, 機械的, 手続き的)アプローチの後ろ盾がないと根付かないですね.
- 縦軸について
  - 世界全ての扱えるわけでないので, どっかで線引きが必要です.
  - ソフトウェア工学の一分野というのは案外妥当な線引きかもしれません.

5

目次

- セキュリティ要求の重要性
- 3つの代表的なユースケース拡張
  - Misuse Cases
  - Abuse Cases
  - Security Use Cases
- 自身の関係する取り組み
  - TOPSEプロジェクト (NII)
  - PORTAM (要求と脅威のトレードオフツール)
  - GANECCO (ドメイン知識利用法)

6

セキュリティ要求の重要性

- 悪い人とかダメな人が多すぎ
  - 心も頭と正しくしないと無駄な出費が・・・・
  - 教育者としては心と頭を正しい人を増やす努力が必要かと・・・
- 典型的なセキュリティの視点
  - Confidentiality 秘密を暴露すんな.
  - Availability 邪魔すんな.
  - Integrity 壊すな.

7

セキュリティ要求をどう決めるか？

- 形式手法やセキュリティモデルは有効であろう.
- しかし, (セキュリティ専門家でない)ステークホルダにもわかり易い説明・提示法が必要.

- ユースケースを利用した獲得・分析法が有効.

8

代表的なユースケースの拡張法

- Abuse Cases
  - by J. McDermott 他 (1999～)
  - 単に悪意あるアクター，処理を別途書く.
- Misuse Cases
  - by G. Sindre 他 (2000～)
  - 悪い処理と良い処理の関係．脅威と軽減
- Security Cases
  - by D. Firesmith (2003～)
  - Security use case を明示化.
  - 再利用に着目.

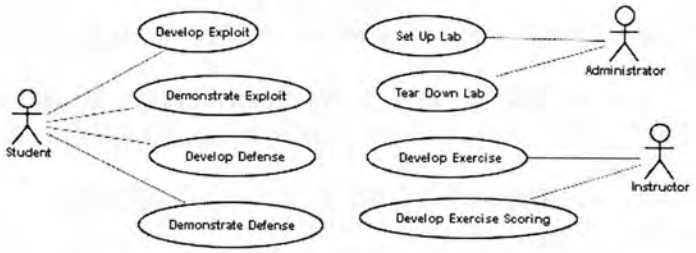
9

Abuse Casesの特徴

- ステレオタイプを含め文法的な拡張は行わない.
- 通常ユースケース図とアブ・ユースケース図は完全に分けて書く.
- アブ・ユースケース図のアクターに関しては，資源，技能レベル，目的を詳細に書いてもよい.

10

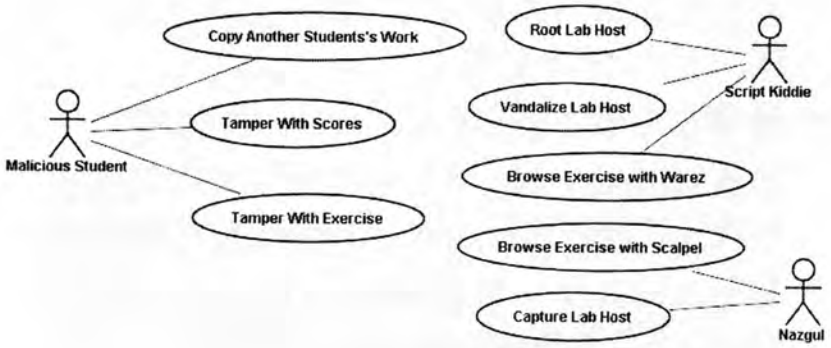
通常のユースケース



セキュリティ教育のWBTらしい.

11

アブユースケース



12

Script Kiddieの仕様

Script Kiddie

[資源] 通常一人で作業を行うが，他のScript Kiddieと情報交換をする場合もある．
必要なハードウェア，ソフトウェア，インターネット接続を有している．
これらは個人で購入したものか，もしくは職場等の資源を悪用する場合もある．
本分析でのScript Kiddieは常時攻撃を行っているものと想定する．
[技能レベル]
高くない．他者が提供するツールや技術の利用法を知っているだけである．
[目的]
暴力行為や窃盗そのものを目的としており，中には自分の腕前を自慢する目的の者もいる．

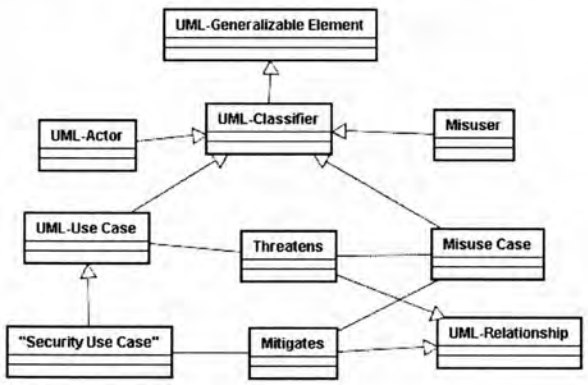
13

Misuse Casesの特徴

- 害となるユースケース，害をなすアクターを構文的に分けて書く．
- 通常なものと害なものを1つの図に書く．
- あるMisuse Caseがユースケースに脅威を与えることを threaten という関連で仕様化．
- あるユースケースがMisuse Caseの脅威軽減となることをmitigateという関連で仕様化．
  - コレはセキュリティケース(後述)に相当する．

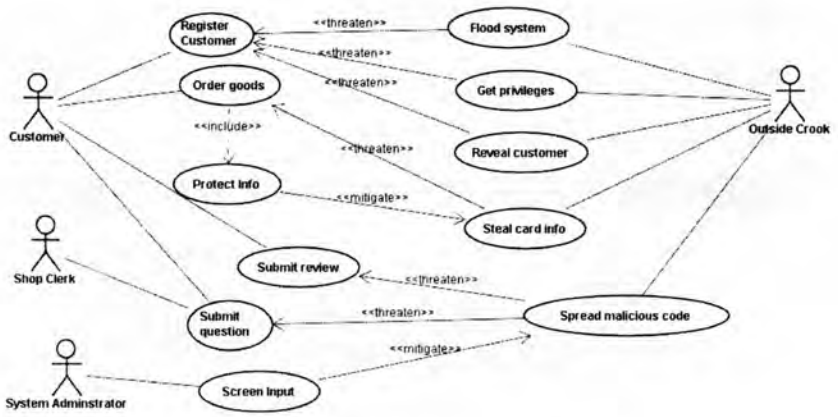
14

メタモデル (データ構造)



15

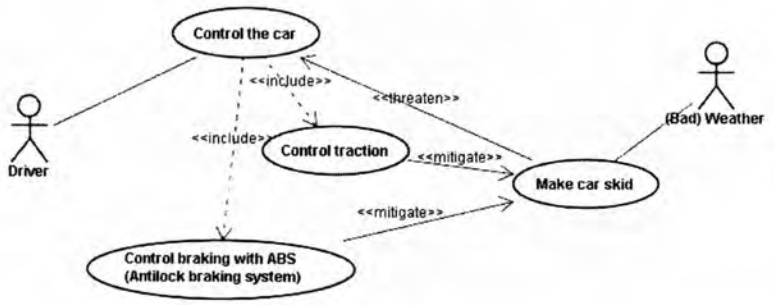
記述例



電子ショップの例

16

安全要求についても対応可能



17

ユースケース記述について

- 軽量化記法と，広範囲記法がある．
- 軽量化記法: 通常ユースケース記述に「脅威」項目を付加したもの．
- 広範囲機能: ミスユースケース主体で書く．

18

軽量化記法の例 1/2

名前: 顧客登録
概要: 顧客は氏名，住所，メールアドレス，電話番号を与え，eショップに顧客登録する.
正常パス: bp-1. 顧客は「登録」を選択する.
bp-2. システムはフォームを提示する.
bp-3. 顧客はフォームを埋めて，送信する.
bp-4. システムは登録完了を知らせ，顧客の参照番号を返す.
代替パス: (略)
例外パス: E1. bp-3において顧客が必須情報を埋め忘れた場合，
より詳細な情報を示しbp-3に戻る.
E2. bp-3において既登録ユーザーの情報と一致した場合，
システムはユーザーに対して既に登録されているので，
現登録作業は破棄される旨と提示する.
そして，本ユースケースを終了する.

19

軽量化記法の例 2/2

仮定: (略)
事前条件: (略)
事後条件: 顧客は登録され，新規に連絡情報等を与えなくてもeショップで
商品を注文することができるようになる.
脅威: T1: 顧客が自身の本名，本住所でなく，仮(偽り)のもので登録しようとする.
起こりうる事態は以下の通り.
T1-1. 実在しない人物が登録されてしまう.
T1-2. 実在人物が本人の意に反して登録されてしまう.
T1-3. 既に登録されている人物が誰かを第三者がチェックできてしまう(E2を用いて).

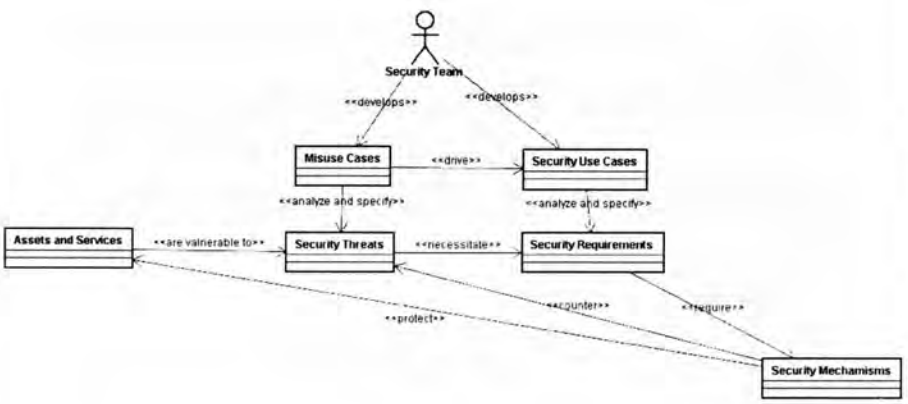
20

Security Casesの特徴

- 基本的なアイディアはMisuse Casesと同じ.
- しかし，セキュリティ面の概念と要求との関係をよく整理している(後述).
- 特に以下の二点が特徴.
  - 実現機構(パスワード認証等)と要求(識別，認証，認定を必要とする等)の区別を明確化.
  - セキュリティ要求は比較的アプリによらず再利用しやすいことを主張.

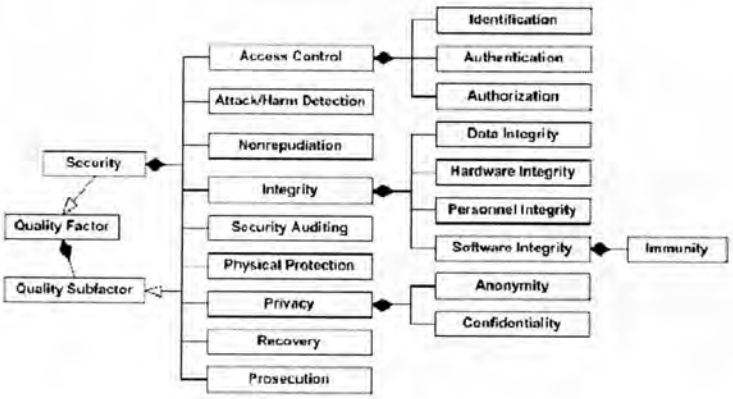
21

メタモデル (データ構造)



22

Security Factorの分類



23

参考文献 (Misuse Case系)

1. John McDermott and Chris Fox. **Using Abuse Case Models for Security Requirements Analysis**. In 15th Annual Computer Security Applications Conference (ACSAC'99), pp. 55-64, Phoenix, Arizona, Dec. 1999.
2. J. McDermott. **Abuse-Case-Based Assurance Arguments.** In 17th Annual Computer Security Applications Conference (ACSAC'01), pp. 366-374, New Orleans, Lousiana, Dec. 2001.
3. G. Sindre and A. L. Opdahl. **Eliciting Security Requirements by Misuse Cases.** In 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-37'00), pp. 120-131, Nov. 2000.
4. Guttorm Sindre and Andreas L. Opdahl. **Eliciting security requirements with misuse cases.** Requirements Engineering, Vol. 10, No. 1, pp. 34 - 44, Jan. 2005.
5. Ian Alexander. Misuse Cases: **Use Cases with Hostile Intent**. IEEE Software, Vol. 20, No. 1, pp. 58-66, Jan./Feb. 2003.
6. Donald Firesmith. **Security Use Cases**. Journal of Object Technology, Vol. 2, No. 3, pp. 53-64, May-Jun. 2003.
7. Donald Firesmith. **Specifying Reusable Security Requirements**. Journal of Object Technology, Vol. 3, No. 1, pp. 61-75, Jan.-Feb. 2004.

24

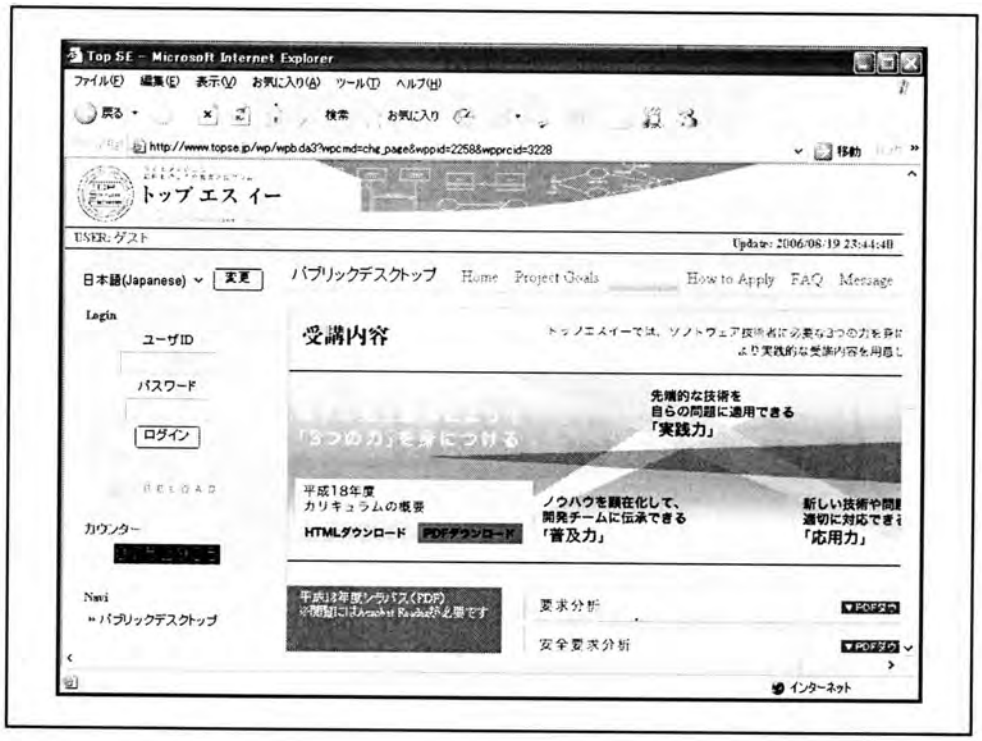
自身の活動と
セキュリティ要求との関係

25

TOPSE

- NII本位田先生がプロデュースする社会人教育プロジェクト.
- CSの素養を活かして開発支援ツールを使いこなすスーパーSEを養成する (らしい)
- 科目の中に「要求分析」「安全要求分析」というのがあり，海谷もお手伝いすることに(割と突然)なってしまった.
- 毎年6月に受講生を公募しているらしいので，来年度は是非お試しを.

26

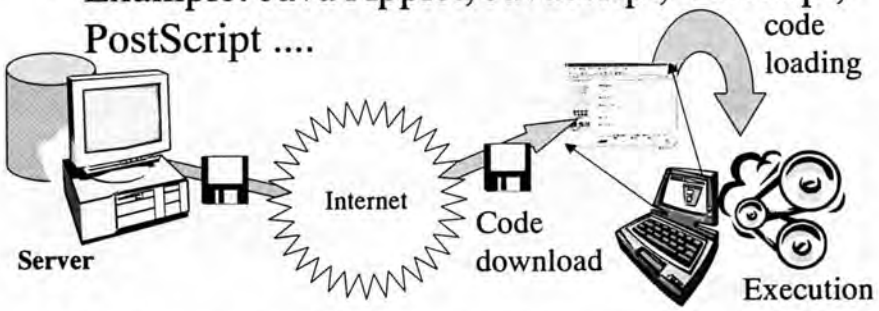


モバイルコードにおける セキュリティ・ポリシィ (PORTAM プロジェクト)

28

What is Mobile Code?

- Example: Java Applet, Javascript, VBscript, PostScript ....



- We do not deal with autonomous mobile codes in our current research.

29

利点と欠点

- 利点
  - 機能の動的かつ柔軟な変更に対応可能.
  - ケースバイケースかつオンザフライで品質や価格の違うソフトウェア部品の入れ替えを可能とする.
  - クライアント側で大きなストレージを必要としない.
- 欠点
  - アセット(資産)に対する保護を適切に行うことが困難.
  - 要求達成の妥協が時には必要.
  - 脅威を黙認することも時には必要.

30

PORTAMツールの概要

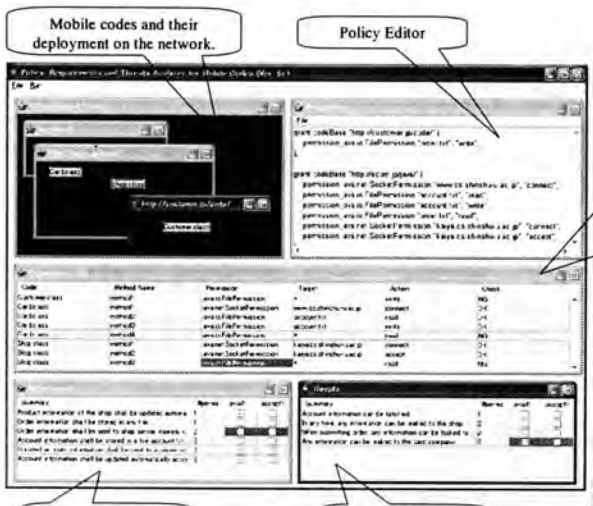
- Who are the users of this tool?
  - Users of mobile code applications.
  - Application integrators.
  - Students learning mobile code security.
- When and where is this tool used?
  - Before running a mobile code application.
  - On the client side.
- Why is this tool required?
  - Potential risks of mobile codes.
  - Trade-offs between requirements and threats.
- What does a user do with this tool?
  - To confirm satisfaction of requirements.
  - To identify threats to be tolerated.
  - To define security policy.

- How to use this tool typically?

1. Describe requirements for an application.
2. Select mobile codes to be used in the application.
3. Identify security related methods used in each requirements.
4. Define policy granting methods used in all requirements.
5. Explore threats under the granted methods.
6. Identify methods causing each threat.
7. Update policy

to abandon some requirements and/or
to tolerate some threats.

31



List of methods and their status under the policy.

Code	the place where the method is embedded.
Permission	the type of permission required by the method.
Target, action	Permisson specific attributes
Check	whether the method can be executed or not.

	Requirements	Threats
white	Satisfied	Avoided
Yellow	Abandoned	Tolerated
Pink	Unsatisfied but not abandoned	Unavoided but not tolerated

Case Study in a class room in winter 2005.

Objective: to confirm how much our tool contribute users to find threats effectively and accurately.
To confirm the improvement of understanding security.
Metrics: the amount of fatal threats found and the amount of errors in finding threats.
Changes of awareness of students.
Methods: comparative experiment with or without our tool.
Questionnaires.
Subjects: 20 Undergraduate School Students.

Results: A little bit better results about both effectiveness and efficiency, but the differences were not statistically significant.
Understanding of students seemed to be improved.

32

既存システム特性を利用した要求定義 (GANECCOプロジェクト)

33

研究背景

- まったく新規のソフトウェアシステムを作ることは稀.
- 実際の開発では，すでに存在する既存のシステムをよく分析した上で開発を行う(ほうが良い).
  - 当然あるべき特性を識別しやすい.
  - 既存との差別化を打ち出しやすい.

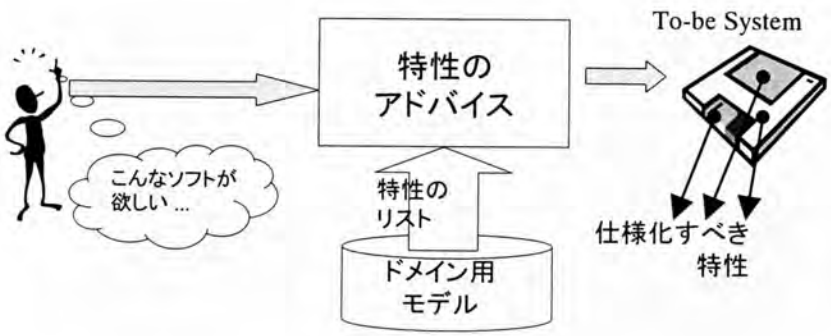
34

研究目的

- 既存類似システムの情報を有効利用するためのデータモデルの定義.
- 実際にそのようなデータベースを作成する手順の明確化.
- 作成したデータ(ドメイン知識)に基づく要求定義の加速と高品質化.

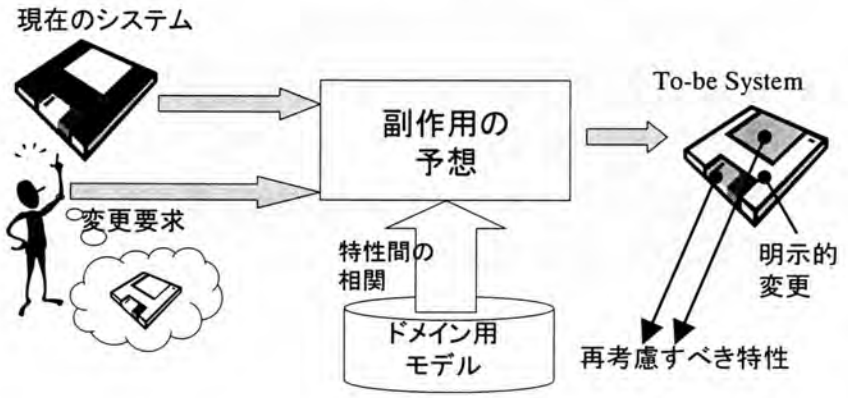
35

支援1 当然定義すべき部分を指示



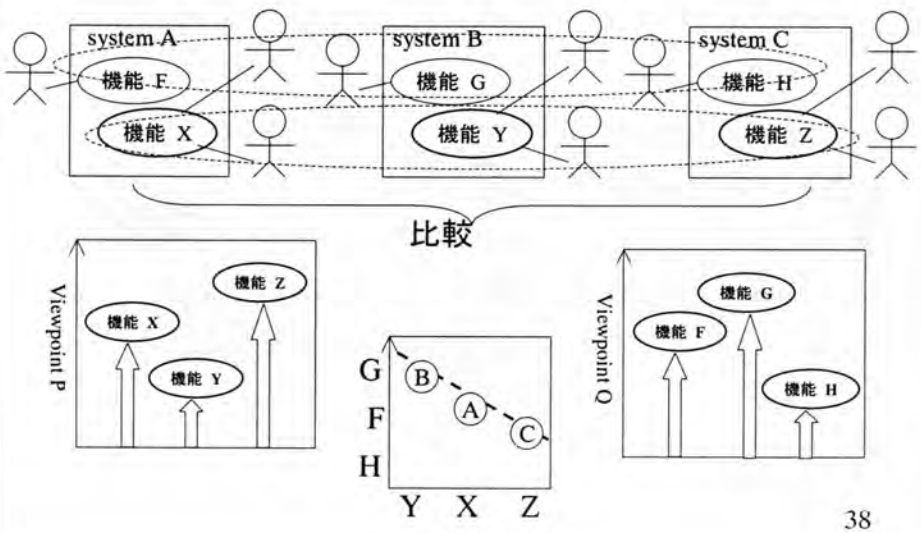
36

支援2 変更による副作用を助言



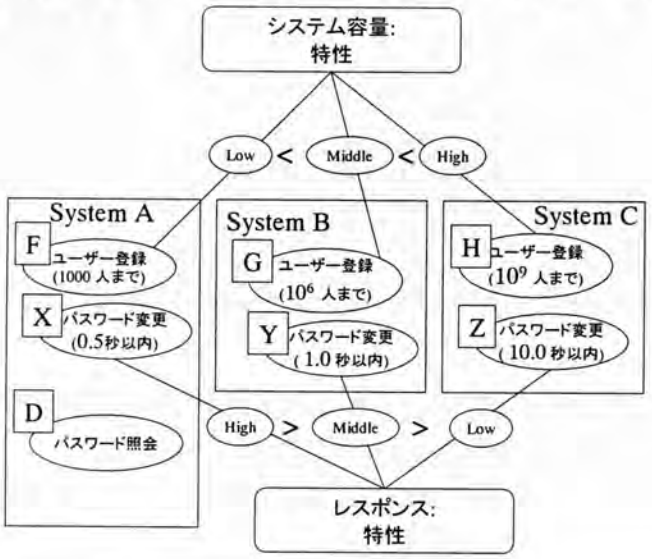
37

ドメイン知識構築の基本方針



38

例: 容量とレスポンスのトレードオフ



39

参考文献等 (手前味噌系)

1. TOPSEページ, 特に安全性要求分析. http://www.topse.jp/
2. Haruhiko Kaiya, Kouta Sasaki, Yasunori Maebashi, and Kenji Kaijiri. **Trade-off Analysis between Security Policies for Java Mobile Codes and Requirements for Java Application**. In 11th IEEE International Requirements Engineering Conference, pp. 357-358, Monterey Bay, California, Sep. 2003.
3. Haruhiko Kaiya, Kouta Sasaki and Kenji Kaijiri. **PORTAM: Policy, Requirements and Threats Analyzer for Mobile Code Application** . QSIC2006 (Sixth International Conference On Quality Software). October 26-28, 2006. Beijing, China.
4. Haruhiko Kaiya, Akira Osada, and Kenji Kaijiri. **Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems**. In Proceedings of 12th IEEE International Requirements Engineering Conference, pp. 112-121, Sep. 2004.
5. Akira Osada, Daigo Ozawa, Haruhiko Kaiya, and Kenji Kaijiri. **Modeling Software Characteristics and Their Correlations in A Specific Domain by Comparing Existing Similar Systems.** In Kai-Yuan Cai, Atsushi Ohnishi, and M. F. Lau, editors, QSIC 2005, Proceedings of The 5th International Conference on Quality Software, pp. 215-222, Melbourne, Australia, Sep. 2005. IEEE Computer Society.
6. 海谷のホームページ http://www.cs.shinshu-u.ac.jp/~kaiya/ ただし8月は休止中.

40

おわりに

- ユースケースを利用してセキュリティ要求分析を行う手法を紹介しました.
- 私自身の取り組みとして上記に関係のある成果をいくつか御紹介しました.

- 多分, セキュリティは比較的取り組み易い分野かと思います.
- 業務固有の部分に関する分析をどう行うかは今後の課題かと思います.

41

SEA Forum August

超上流から攻めるIT化の勘所

2006年 8月21日（月）

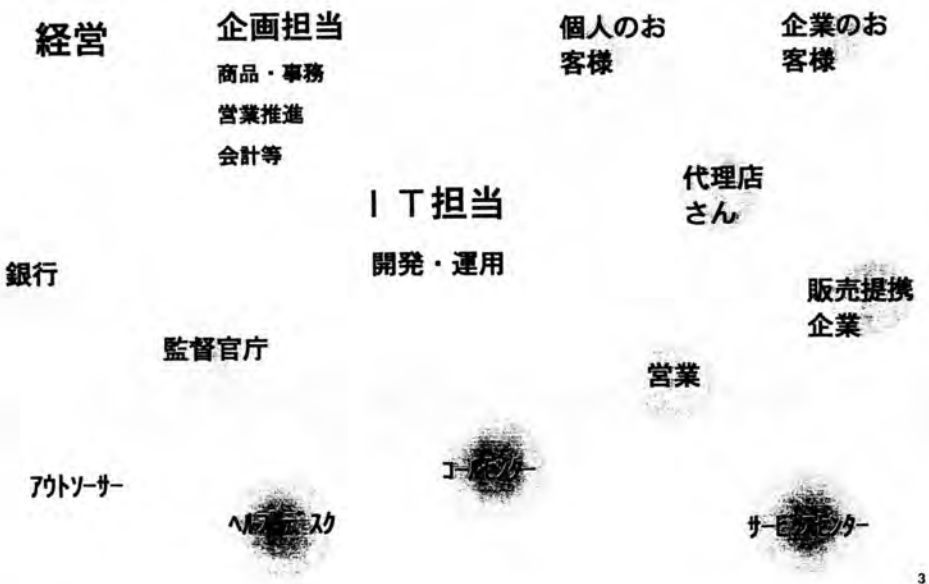
情報処理推進機構・SEC リサーチフェロー
東京海上日動システムズ技術顧問
株式会社アイネス・金融システム本部・副本部長
菊島 靖弘

0

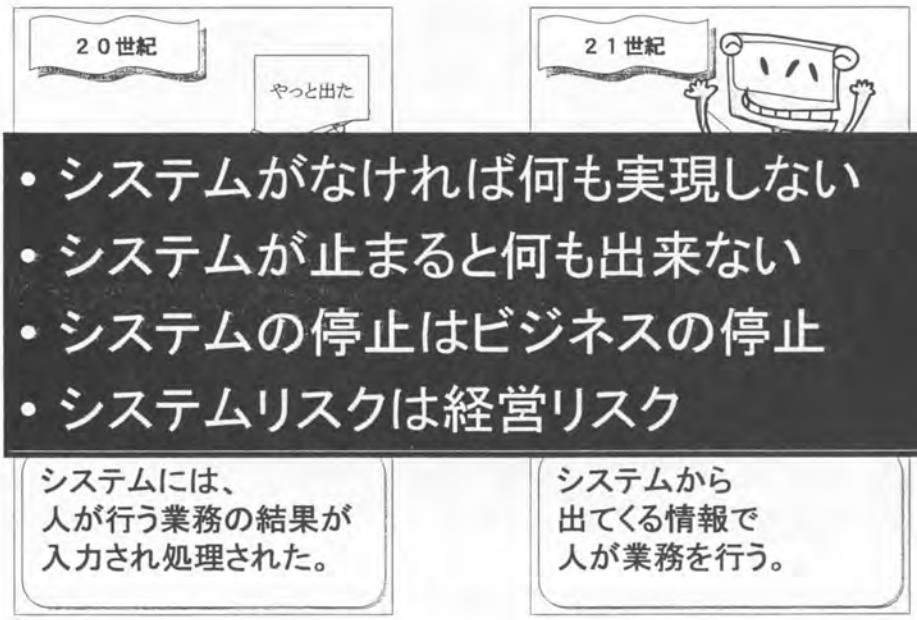
システムがなければ何も実現しない

1. 新しい技術は、新しいビジネスを可能にする

2. ステークホルダーの増加



3. システムは業務基盤となった



- システムがなければ何も実現しない
- システムが止まると何も出来ない
- システムの停止はビジネスの停止
- システムリスクは経営リスク

システムには、
人が行う業務の結果が
入力され処理された。

システムから
出てくる情報で
人が業務を行う。

4

4. 新しい役割

「僕つかう人、わたし作る人」の時代ではない！

かつては「発注者」であり「使用者・評価者」

業務・商品設計者

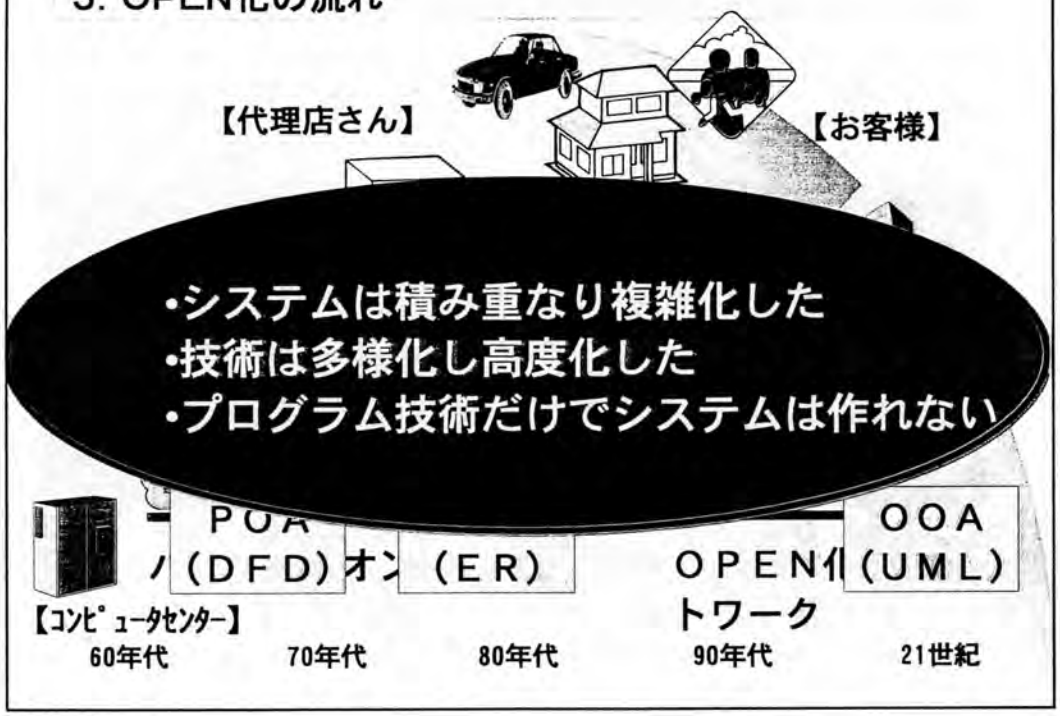
ＩＴ部門

かつては「受注者」

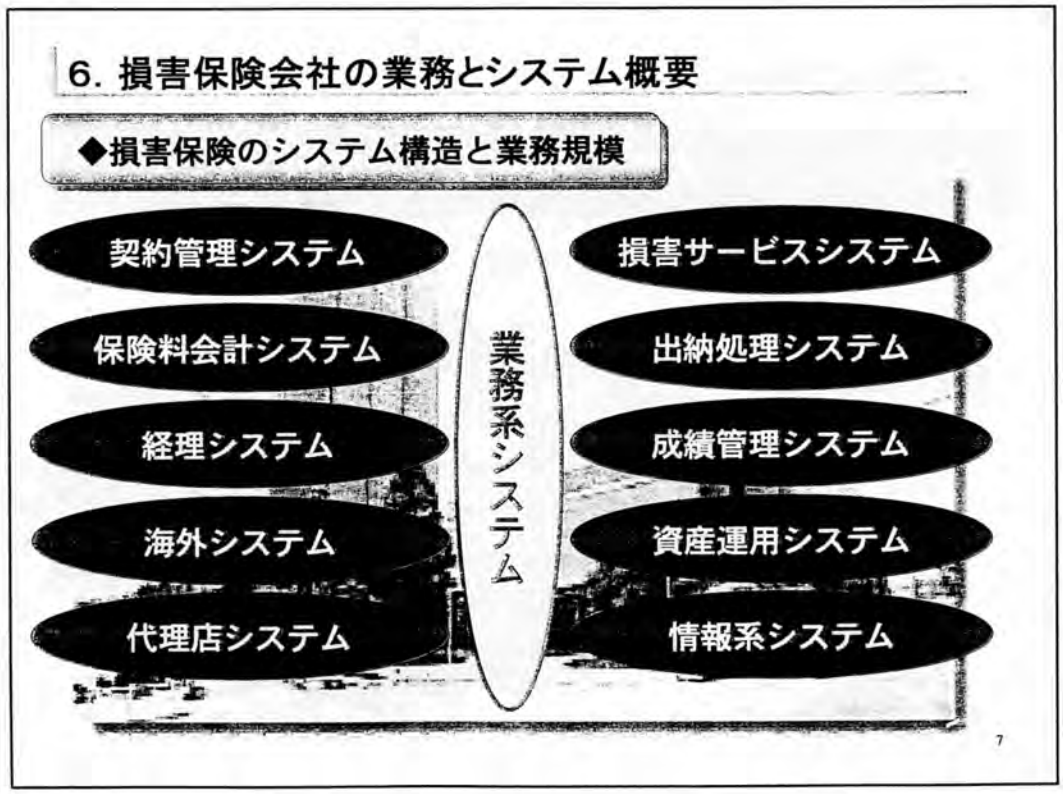
今やマーケットが利用者であり、評価者。
両部門が新しいビジネスモデルの提供者として、共同し、業務・事務設計、システム設計・開発を行う体制に！

5

5. OPEN化の流れ

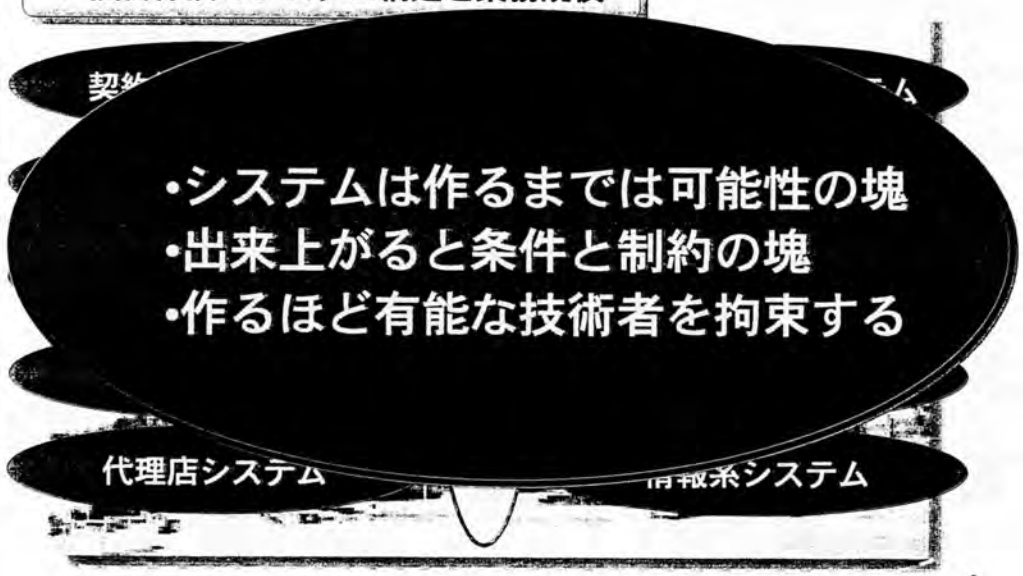


6. 損害保険会社の業務とシステム概要



6. 損害保険会社の業務とシステム概要

◆損害保険のシステム構造と業務規模



8

ちょっとブレイク＊整理してみましょう＊

- システムは企業の外に飛び出し社会基盤化した。
- 利用者が増加し、要求が多様化した
- システムがなければ何もできなくなった
- システムを作ってという人が作る側になった。

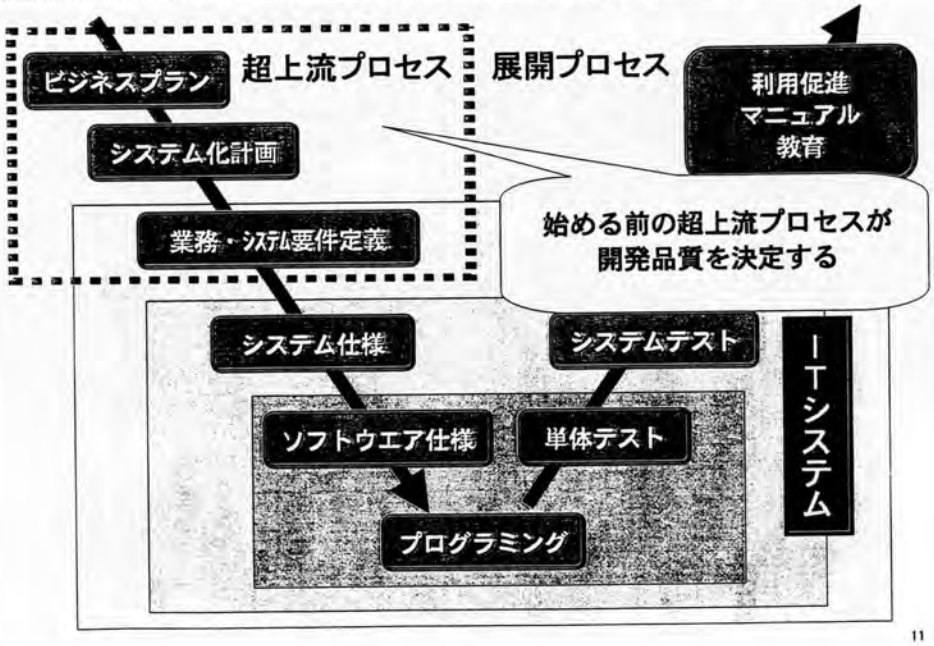
9

2

要件定義がシステム開発力強化の鍵

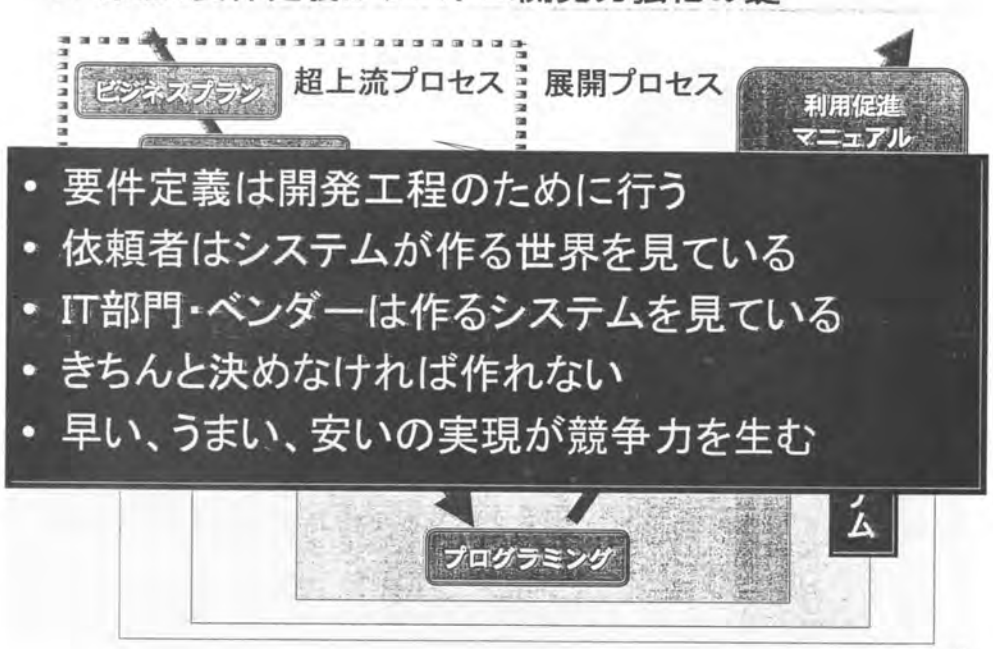
10

7. 原点:要件定義がシステム開発力強化の鍵



11

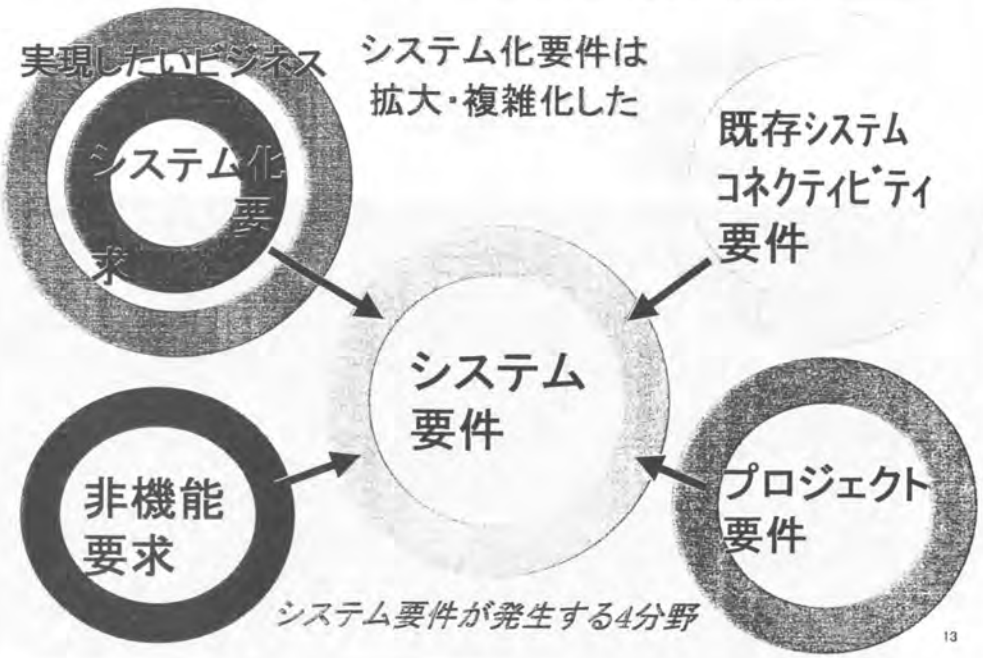
7. 原点：要件定義がシステム開発力強化の鍵



- 要件定義は開発工程のために行う
- 依頼者はシステムが作る世界を見ている
- IT部門・ベンダーは作るシステムを見ている
- きちんと決めなければ作れない
- 早い、うまい、安いの実現が競争力を生む

12

8. 要求から要件へ（要件定義が難しくなった）



システム要件が発生する4分野

13

8. 要求から要件へ

ビジネスフィールド(1)

1.ビジネス構想の不完全さ対応を織り込む
（詳細不足、曖昧さ、目的不明、前提不明）
2.実現したい世界の構想は往々にして、[illegible]
一方的、個別・部分最適であり、結果として、みなし、決め付け、
思い込み、勘違い、漏れ、誤りを含むことを評価する
3.システム化要求の内容は未承認であることが多い、承認
レベルをおさえて進める

14

8. 要求から要件へ

ビジネスフィールド(2)

1.システム化要求の[illegible]をおさえる
2.システム化要求にかかる、全ての
意見反映レベルをおさえる
3.発注者は利用者でなくなってきていることを考慮する
（マーケットの声が聞こえていない）
4.特に、担当者が勝手にこうなると思い込んでいると
判断される要求は吟味・管理する
5.[illegible]

8. 要求から要件へ

1.かってに、そうなるとおもいこんでいるところの吟味
（機構、承認、体制、事務の仕組、使い勝手、背景不明作業など　　要は書いていないこと）

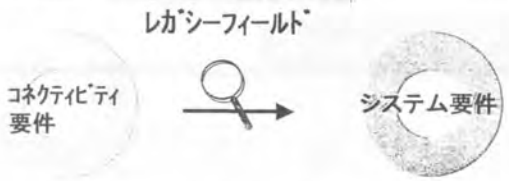
2. レスポンス、キャパシティ、セキュリティ、トー
ラビリティ、メンテナンスビリティ

3. 有事対応、障害時対応、コンテンジェンシープラン

4.環境（何台までしかつながらない、どこでしか出来ない）、
人（誰しか出来ない、同時に出来ない等）、
物、金（後年度負担）、24時間365日等

5.特許対応、法律・認可対応

8. 要求から要件へ



1.現行システムとその利用、運用

（前提、内容、事情、制約、研修、運用、特殊運用、担当、契約、過去のイベント対応等）

2.アウトプット活用内容（追加加工、データ二次活用）

3. アウトプットステークホルダー対応

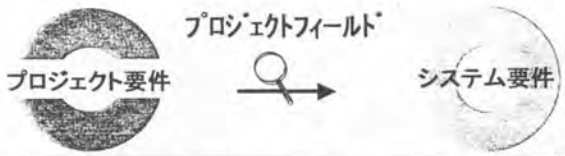
（代理店、客先（個人、企業）、社員、利用管理者、監督官庁、コールセンター）

4. 関連システム（含むIF、インターフェイス、内容、事情、制約、運用等）

5. インフラ（キャパシティ、OSバージョン、ネットワーク、認証契約、アウトソース関連等）

17

8. 要求から要件へ



1. 受発注者、マーケット、客先要望事情等、ステークホルダーの個別事情
2. 同時期イベント
3. 社内インフラ改定、[illegible]、制度改定、機構改革等
4. [illegible]
5. 社外利用インフラ改定(郵便番号等)、料金改定等外部環境条件
6. 開発力確保、パートナー・アウトソーサー・客先の開発力
7. 開発中・展開時有事対応(地震等天災、展開季節、利用パッケージ
[illegible]同時開発・プログラム共有システムの有無

18

9. 発注者は自分の欲しいものがわからない？

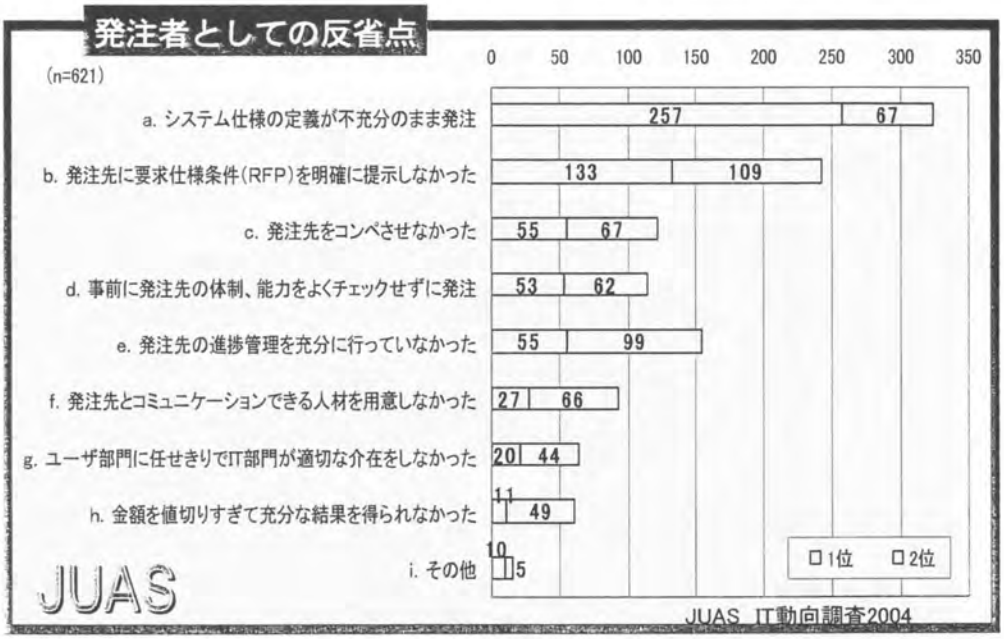
要求仕様書(RFP)における役割分担

- ラフな要求仕様だけで、残りはベンダに書いてもらっている企業や、ベンダにすべて書いてもらっている企業が、全体の半数

JUAS IT動向調査2004

9. 発注者は自分の欲しいものがわからない？

発注者としての反省点



JUAS

JUAS IT動向調査2004

3

要件定義力をつけ、発注力を強化しよう

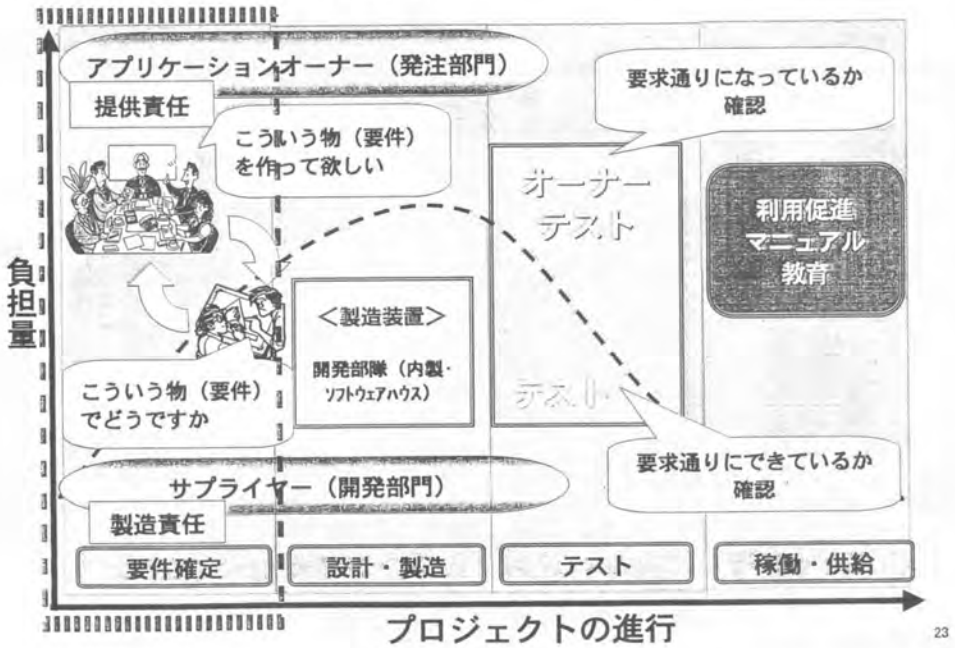
21

10. 経営の課題と役割

要件定義力をつけるためには

- 企画発注部門のIT力強化
  超上流工程の強化
- IT部門の強化
  システムがビジネスに命を吹き込む
- 選択と集中
  政治的選択から戦略的選択へ
- 品質確保は社会的責任

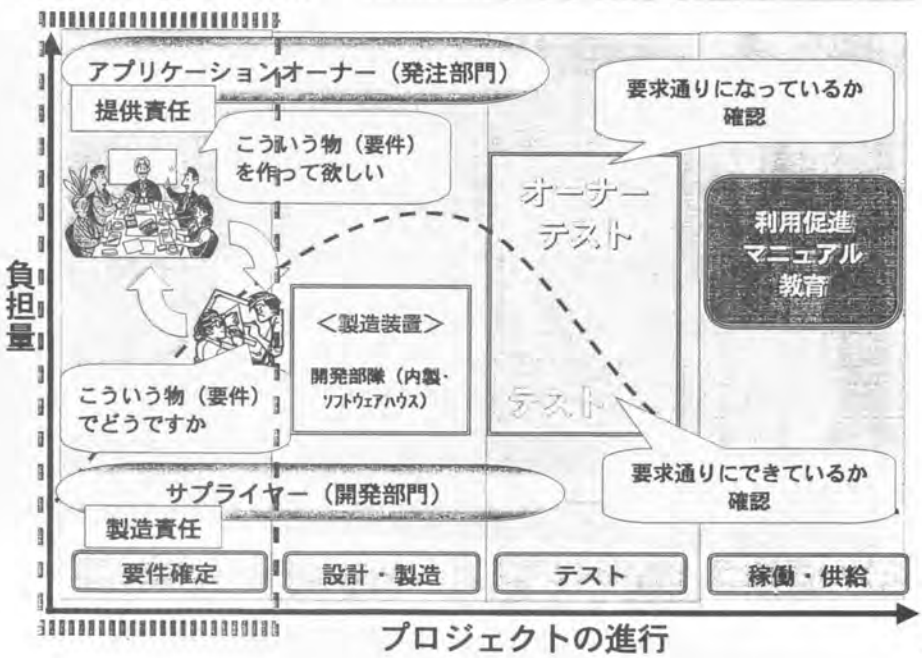
11. 企画業務部門の役割



23

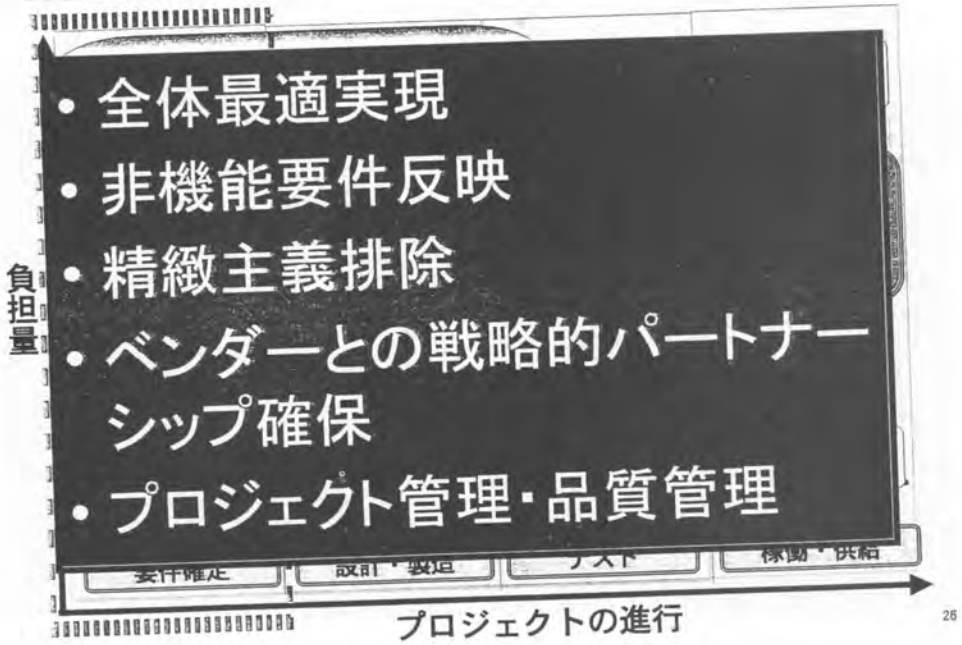
11. 企画業務部門の役割

12. IT部門の役割



25

12. IT部門の役割



26

13. 超上流工程を進めるポイント

- ポイント1：最低限やりたいことを考える
- ポイント2：システム化ありきで進めない
- ポイント3：優先順位を決めてとりかかる
- ポイント4：シンプルな業務設計を心がける
- ポイント5：死守すべきQCDを決める
- ポイント6：プロジェクト体制の穴を見る
- ポイント7：責任の所在を明確にしておく

出展：ＳＥＣ　ＢＯＯＫＳ
「経営者が参画する要求品質の確保」

27

ありがとうございました
（要件定義力をつけましょう）

付録1．超上流工程の実際

・戦略的中長期案件
・他社対抗新商品
・客先要望対応
・許認可・法制度対応　　等

案件発生 → 打診 → 基本計画策定 → 承認決済 → 開発スタート → 開発計画作成 → プロジェクトレビュー

29

付録1. 超上流工程の実際

・年次計画
・緊急案件
・経営からの指示
・相談　　等

案件発生 → 打診 → 基本計画策定 → 審議・承認・決済 → 開発スタート → 開発計画作成 → プロジェクトレビュー

30

付録1. 超上流工程の実際

・業務企画＋ＩＴ部門共同作業
・プロジェクト必要作業の洗い出し
・大枠のシステム対応内容確定
・ビジネス企画内法としてのＩＴコスト　等

案件発生 → 打診 → 基本計画策定 → 審議・承認・決済 → 開発スタート → 開発計画作成 → プロジェクトレビュー

大きな案件ではここの作業品質が次工程以降に大きな影響をおよぼす

31

付録1. 超上流工程の実際

・情報化委員会による選択と集中
・経営による決済（予算確定）
・緊急案件対応　　等

案件発生 → 打診 → 基本計画策定 → 要請承認決済 → 開発スタート → 開発計画作成 → プロジェクトレビュー

32

付録1. 超上流工程の実際

・プロジェクト登録
・関係ステークホルダー確認
・開発体制構築開始
・ベンダー打診　　等

案件発生 → 打診 → 基本計画策定 → 要請承認決済 → 開発スタート → 開発計画作成 → プロジェクトレビュー

33

付録1. 超上流工程の実際

- 業務要件定義（含むｽﾃｰｸﾎﾙﾀﾞｰﾆｰｽﾞ）
- 非機能要件洗い出し、リスク評価
- 保有システム調査
- 基盤検討
- 開発体制、計画策定　　等

案件発生 → 打診 → 基本計画策定 → 最適構想決定 → 開発スタート → 開発計画作成 → プロジェクトレビュー

大きな案件ではこの作業を基本計画時に先取り実施する必要がある

34

付録2. 選択と集中の仕組み

◆情報化委員会

委員長
情報システム担当役員

事務局

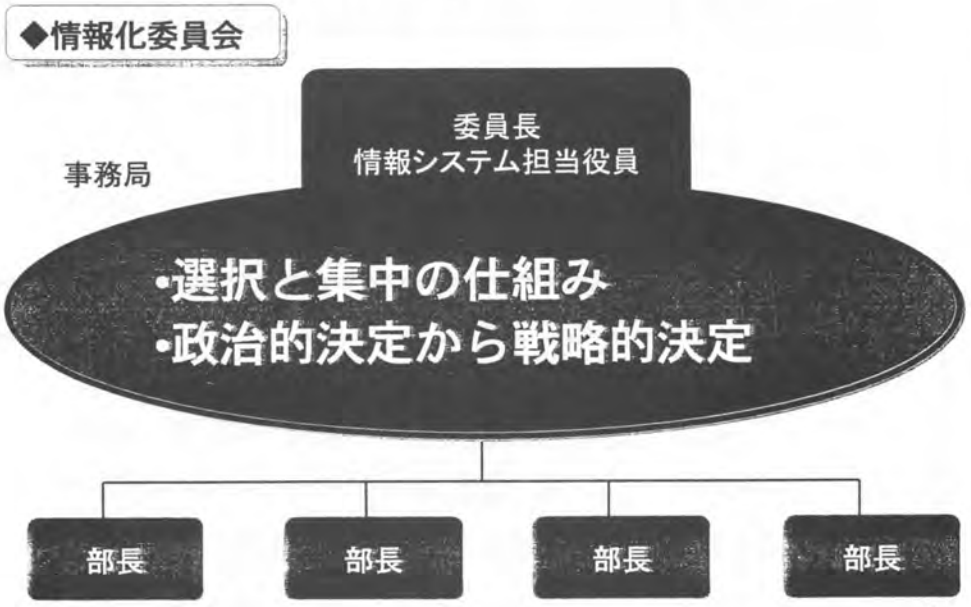
経営企画部
情報システム部

業務サービス部

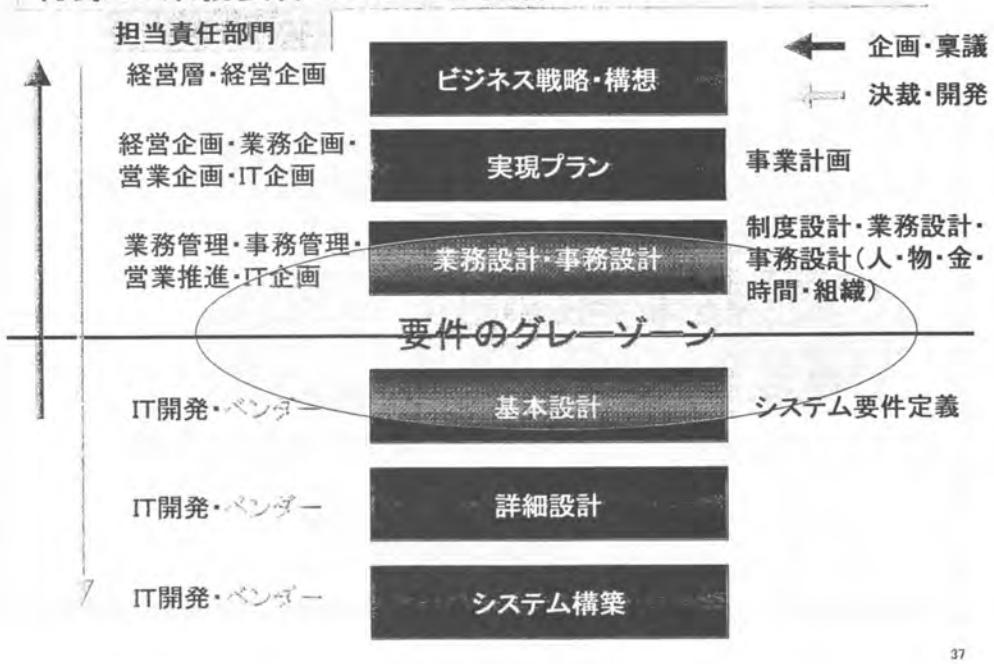
部長　部長　部長　部長

35

付録2. 選択と集中の仕組み



付録3. 業務要件とシステム要件



1

非正常系シナリオ

民生商品での非正常事象の要求仕様抽出

松下電工株式会社／九州工業大学　三瀬

2

目次

1. 組込みシステムに要求される品質
2. 組込みシステムの課題
3. 非正常分析手法の紹介
4. 要求工学の課題の視点

1．組込みシステムに要求される品質　3
・組込みシステムの特徴と安全性要求

対象範囲

民生品を対象とした組込みシステム

不特定多数の人により不特定多数の環境下で使用される

- 多少故障しても何とか使い続ける（寿命末期の安全性）
  - 生活の基盤となっているものなど簡単に機能停止できない
- 間違った使い方、間違った目的で使う（イタズラを含む）
- 極めて僅かでも安全上の問題を起こせない
  - 石油ファンヒータ、ガス湯沸かし器、エレベータ・・・

組込みシステムでは、徹底した安全性が要求される

1．組込みシステムに要求される品質　4
・組込みシステムの安全性とその他の品質

安全であれば、システムは動作しないほうが良い？

トレードオフでは許されない、両立が要求される

異常があれば何でも停止	⇔	出来る限り動作させる
確実に安全性を保てる 障害を起こさない		耐久性が高い 不便にならない（使用性）

⇩

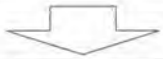
安全は、使用性、効率性など全ての品質とリンクしている。
・利用者の感覚で企業ブランドに対して評価する

EX.　センサーが故障したので、もう動作しません。
→　今どうしても使いたいユーザはどう判断する？
（　メーカの都合は聞かない　）

2．組込みシステムの課題 5
• 何が問題なのか

問題なのは、予測していない状況が存在すること

ソフトウェアは、仕様の通りに動作する。仕様に記述されない状況が発生した場合、どうなるか分からない。



- 障害を予測さえできれば、それへの対策はシステム毎にどのようにでも対策が立てられる。
- 但し、出来る限り早い段階で予測しなければならない。
  - その対策がハードウェアの設計変更を含むことになると、問題は更に大きく、複雑になる。

2．組込みシステムの課題 6
• なぜ予測していない状況が存在するのか

予測しにくい状況を定義することが難しい

「実際に予測できない状況は、非正常現象の連鎖で起きていた。」

例（空想：ばかげた話ですが、シビアな話を出来ないので）

1. 小さい子供がプリンタを開け、手を突っ込む。
2. プリンタは、インク変更モードではないので、ヘッドが触られないように隅へ移動する。
3. 小さい子供の手が挟まる。
4. プリンタは、蓋が開いているので、操作ができない。
5. したがって、プリンタヘッドを動かせない。
6. 手が挟まれているので蓋が出来ない。
7. 子供は泣きながら指を動かす。
8. 近くの電気回路の絶縁被覆部がはがれて子供が感電する！！

2．組込みシステムの課題 7
・障害に対する認識の必要性

障害に至るシナリオとその重大性の認識がないと、
原因の対策に抜けが発生する

EX1.・ 運転中、携帯が鳴る、助手席の書類が崩れる、後ろの同乗者がドッキリするようなことを話す。
- そのとき、前の車が急ブレーキを踏んで停車する。
  - →これだけならば、何の問題もないが．．．
- このようなヒヤリ（＝重大性を認識）経験をすると、人は携帯を運転中モードにする（対策）ようになる。

EX2.・ あるシステムは、開始ボタンと終了ボタンで操作される。
- 誤操作回避のために、2つのボタンの同時押しを受け付けない仕様になっている。
- 障害発生！開始ボタンを押したら、ボタンが戻らなくなった。
  - （その結果）システムの動作を止めることが出来なくなった。
  - 火を消せない、ガスを止められない、機械を止められない。
- 重大性の認識→対策：停止ボタンは、必ず実行される仕様へ。

2．組込みシステムの課題 8
・非正常系について

障害に至るシナリオは、正常の期待しているシステムの特性に対して、状況の逸脱が原因となり障害へのシナリオになる。

正常系： 期待しているシステムの能力とふるまい

非正常系：期待していない正常系からの逸脱

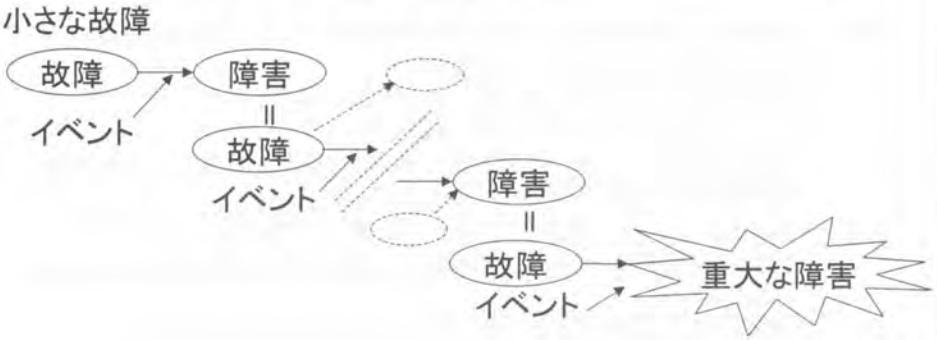
- 正常系
  - 正常（期待する特性）
- 非正常系
  - 準正常（可逆的特性変化）
  - 異常（非可逆的特性変化）

非正常の要因：

人	：	誤操作、誤使用、イタズラ、・・・
環境	：	温度、振動、雑音、雨、風、他の設備、・・・
施工、設置	：	施工不良、施工の劣化・・・
回路機構	：	部品誤動作、部品劣化、部品故障、接触不良・・

2．組込みシステムの課題 9
・非正常系シナリオについて

非正常系シナリオには、重大な障害に至るシナリオも含まれている。



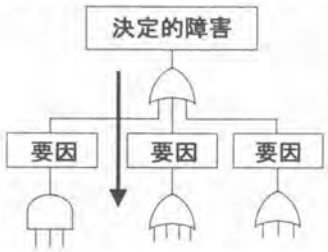
2．組込みシステムの課題 10
・組込みシステムにおける対策すべき課題

重大な障害シナリオを定義する技術は
設計者の経験に依存しており
障害シナリオを定義する手順も定義されていない。

- 対処すべき非正常系現象は、設計者により予測結果のバラツキが大きい。
- レビューを十分に行えない。
- ノウハウが、組織的に蓄積されにくい。

2．組込みシステムの課題 11
・従来からの設計段階の障害分析手法

FTA（**F**ault **T**ree **A**nalysis） FMEA（**F**ailure **M**odes and **E**ffect **A**nalysis）



部品	故障モード	影響
	→	

HAZOP（**Haz**ard and **Op**erability Analysis）

ガイドワード：no, not, more, less, as well as, part of, reverse, other then

分析箇所	考慮する逸脱	原因	起こりうる結果	予防手段	対策
	→	→			

2．組込みシステムの課題 12
・これからの障害分析手法に必要な要素

設計段階での障害分析手法
FTA、FMEA、HAZOP

⇩

従来よりハードウェア中心に適用

⇩

近年高度な情報処理技術が商品へ組み込まれるようになった
システムは極めて多くの状態やイベントを持ち、様々に振舞う

⇩

多くの組合せや状態変化の連鎖によって発生する
障害シナリオへの配慮が必要となった

⇩

しかし、具体的な障害シナリオを記述する手順は明確ではない

3．非正常分析手法例（ESIM）

13

1. 非正常現象の抽出
   - システムの各構成要素で発生すると予想される非正常現象を抽出
   - 構成要素間で発生すると予想される非正常現象を抽
2. 非正常現象と、それへの対処方法、その重要度を分析
3. 対処できていない非正常現象がシステム全体の各構成要素へ与える影響を分析
   →新しい（＝当初は予想できなかった）非正常現象を抽出
4. 2～3を繰り返す→重大な障害を発生させるシナリオを得る
   - どこで非正常現象の連鎖を止めるかを検討する。

ハードウェア再設計、またはソフトウェア設計で
障害シナリオへの対処方法を取り込む

4．要求工学の課題の視点 ・要求工学に対する安全性の扱い

14

- ユビキタス社会に進展していく中で安全が確保できるか。
  - もちろん問題は、安全をベースにした全ての品質
  - 現在想像していないものが、同一ネットワークで接続
  - 社会インフラ（社会基盤として、停止できない、安全確保）
- オフショア開発が進む中で、誰がどのように品質を保証
- IEC61508（機能安全）の動向

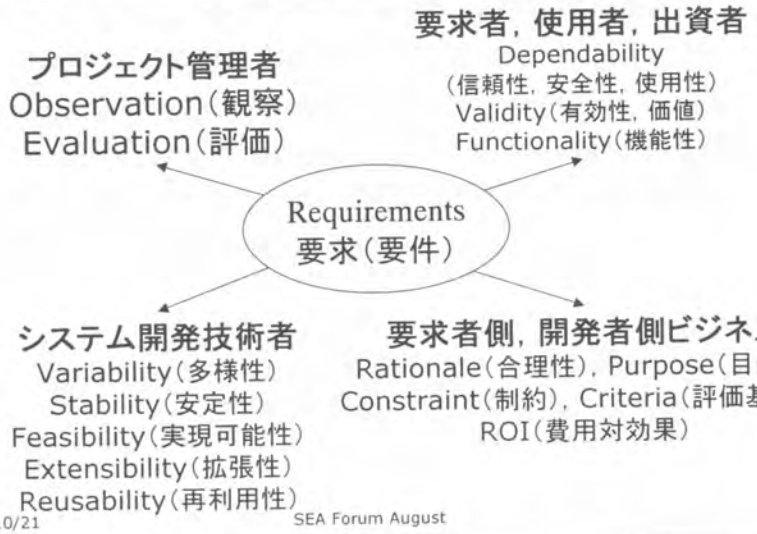
品質の日本ブランドを情報システムに通用させる
ために果すべき要求工学課題は？

4．要求工学の課題の視点　15
・要求工学の課題

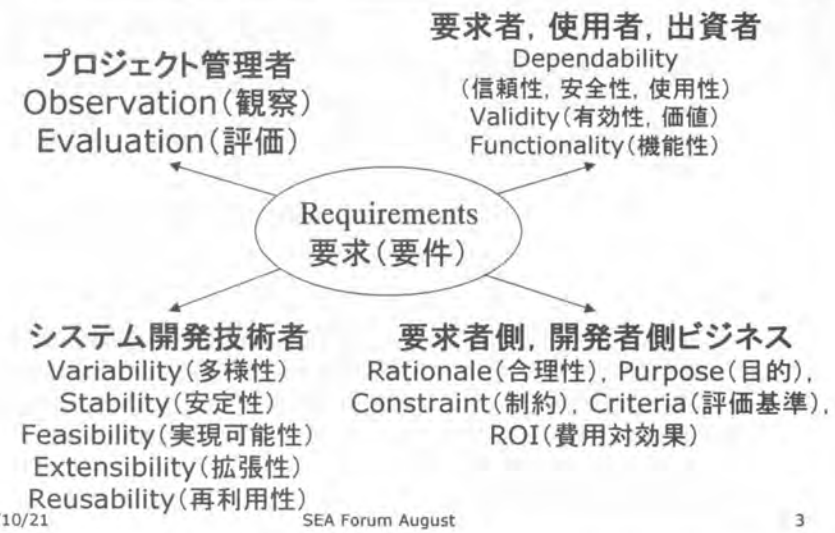
- 超上流分析･･･民生品でも商品企画に必要？
- 情報の高度化と社会インフラ化が進展し、従来の開発業務
  - 全てが、情報処理技術の視点から抑えないといけない。
- 従来の情報処理技術者の役割範囲を超える必要があり、
  - 要求工学のはすべき役割が最も大きい。
- 今、システムの周りで行われている業務の分析から入る必要があるのではないか。
- 従来の手法を否定するのではなく、良い文化を利用しながら拡張していく必要がある。
- 要求仕様の曖昧な定義や未定義な事項が問題
  - それは、個人の問題、書き方の問題、手順の問題などなど
- あるいは、技術進化のひずみを是正していない
- 何を要求として定義すべきか＝本質的要求工学の問題では？

パネル討論：要求工学

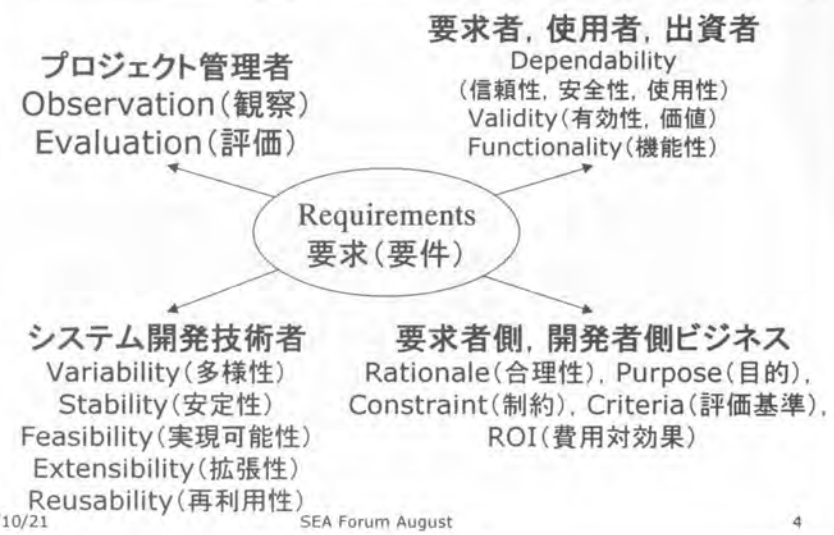
ユースケースを用いた セキュリティ要求分析法



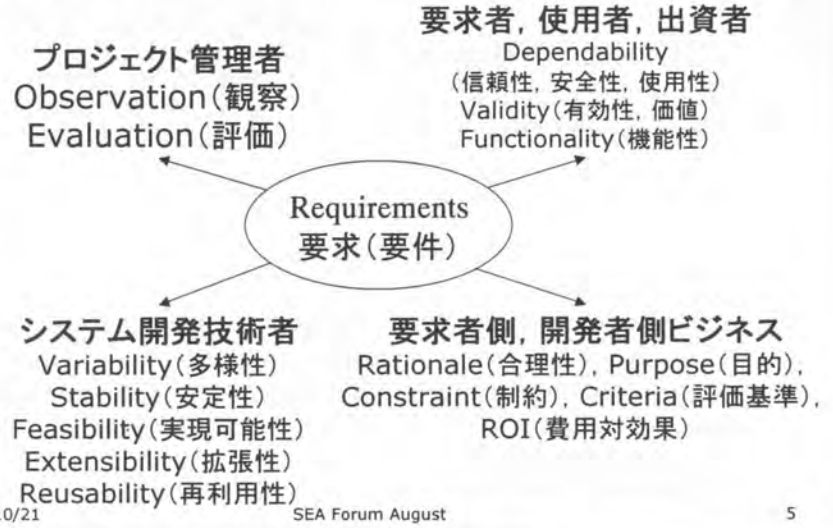
MOYA



非正常系抽出法



質疑



金庫問題

山崎利治: “DW2005のための問題”, ソフトウェア技術者協会, SEAMAIL, Vol.14, No.15(01/2005), pp.51--52.

ご利用に際して

- あなたの物置番号は125です．いま扉は閉まっていて施錠してあります（施錠灯がついています）．
- 物置番号の書かれた物置の扉にはパネルがあり，そこに，施錠灯，0から9までの押ボタン，C，Lの押ボタン，10進数字4桁の液晶表示窓，90度回転する取手が付いています．

扉の開けかた

1. 数字ボタンを1234(施錠鍵といいます）と左から順に押してください．
2. 押した数字を表示窓で確認してください．
3. それが1234であれば施錠灯が消えます（解錠）．
4. そこで扉の取手を時計回りに90度廻して手前に引けば扉が開きます．
5. 数字ボタンを押し間違えたときは，Cボタンを押して，始めからやり直してください（Cボタンを押すと数字窓の数字が消えます）．

扉の閉めかたと施錠

1. 扉を閉め取っ手を逆時計回りに90度廻します(機械的な施錠).
2. Lボタンを押します(電子的な施錠).
3. ここで施錠灯が点きます(直前に扉を開けたときのあるいは,次項によって変更した鍵による施錠).
4. その後は解錠手続きをしない限り,取手を廻しても引いても扉は開かないはずです.

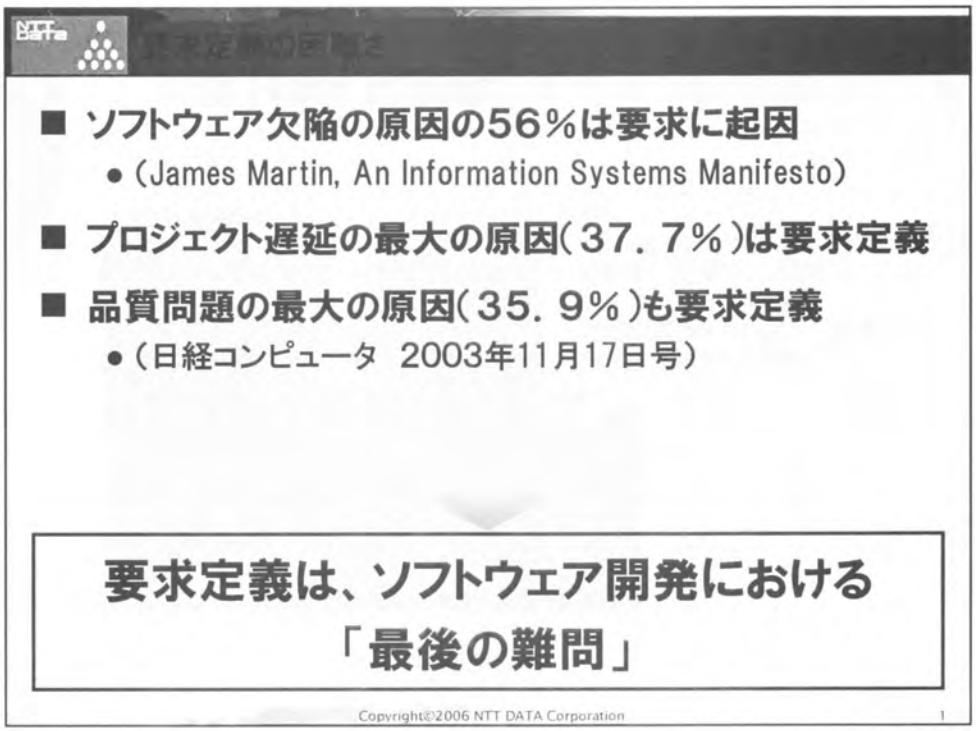
The Real World

バッテリーが死んで“友軍砲火”死 タリバンを狙ったはずが米軍兵士が. .

By Vernon Loeb(Washington Post Staff Writer). Sunday, March 24, 2002; Page A21

- タリバンを砲撃するためにセットされた米軍の衛星誘導装置付き爆弾が，タリバンではなく自軍兵士23名を死傷させた.
- 経過:
  - 空軍の戦闘用制御装置には精密な軽量GPS受信機が使われていた.
  - 制御装置に攻撃目標地点を設定後，GPS受信機のバッテリが死んだ.
  - 兵士はバッテリを交換後，爆弾を発射した.
  - その爆弾はタリバンではなく，発射地点へ戻ってきた.
- 仕様:
  - 受信機のバッテリが交換されたら，制御装置は目標値として現在地を設定する.
- 原因:兵士が仕様を知らなかったこと？


NTT DATA
要求の質的向上を狙う
モデリング方法論MOYA
Corporation


NTT DATA
■ ソフトウェア欠陥の原因の56%は要求に起因
● (James Martin, An Information Systems Manifesto)
■ プロジェクト遅延の最大の原因(37.7%)は要求定義
■ 品質問題の最大の原因(35.9%)も要求定義
● (日経コンピュータ 2003年11月17日号)
要求定義は、ソフトウェア開発における
「最後の難問」
Copyright©2006 NTT DATA Corporation
1

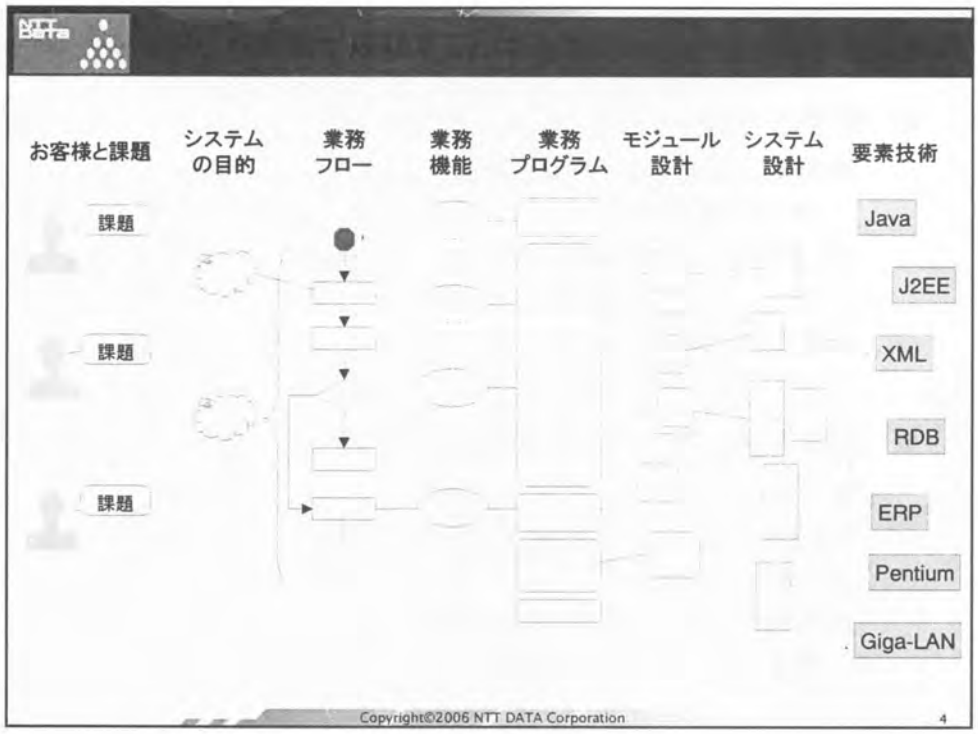
どうすれば仕様は安定するの？

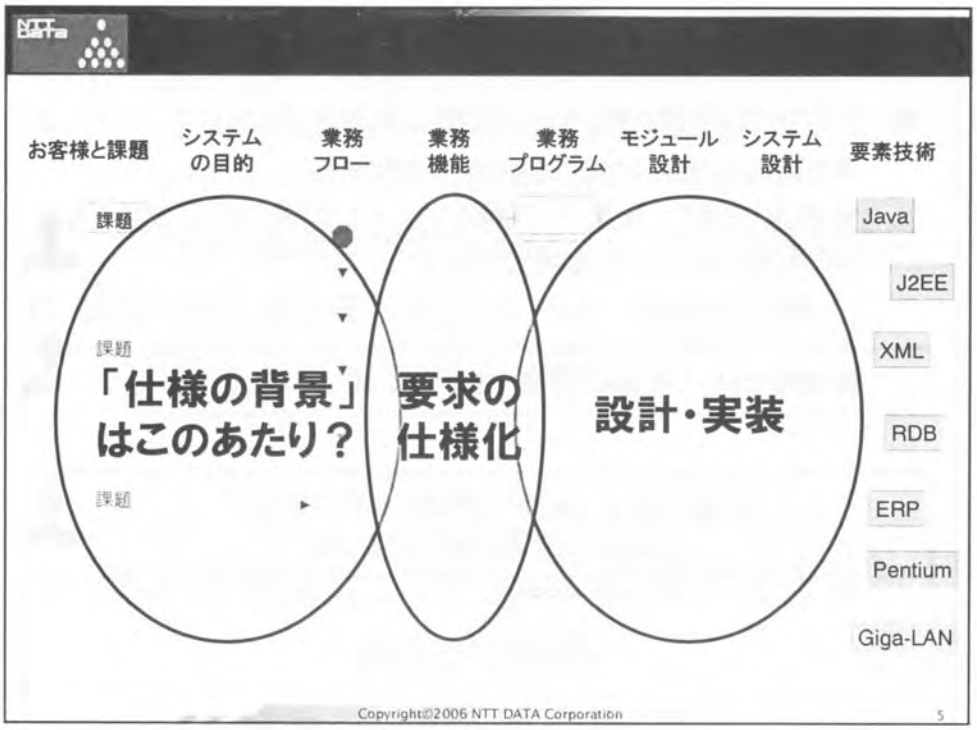
- 頑張って仕様を精緻かつ大量に書く
  - 目次、様式類を緻密に整備する(IEEE 830-1998、Volere Template、など)
  - しかし、いくら書いても書ききれない。たくさん書くと検証できないし、維持もできない...
  - 大量にドキュメントを用意するとかえって要求は不安定に
- 仕様を書くのは諦め、作ってみて仕様を確認する
  - プロトタイピング、イタレーション...
  - しかし、作っても作っても安定しない。作り直しの費用と時間に耐えられない...
  - 簡単に変更できるものはよいが、アーキテクチャに大きな影響を与える変更は不可能

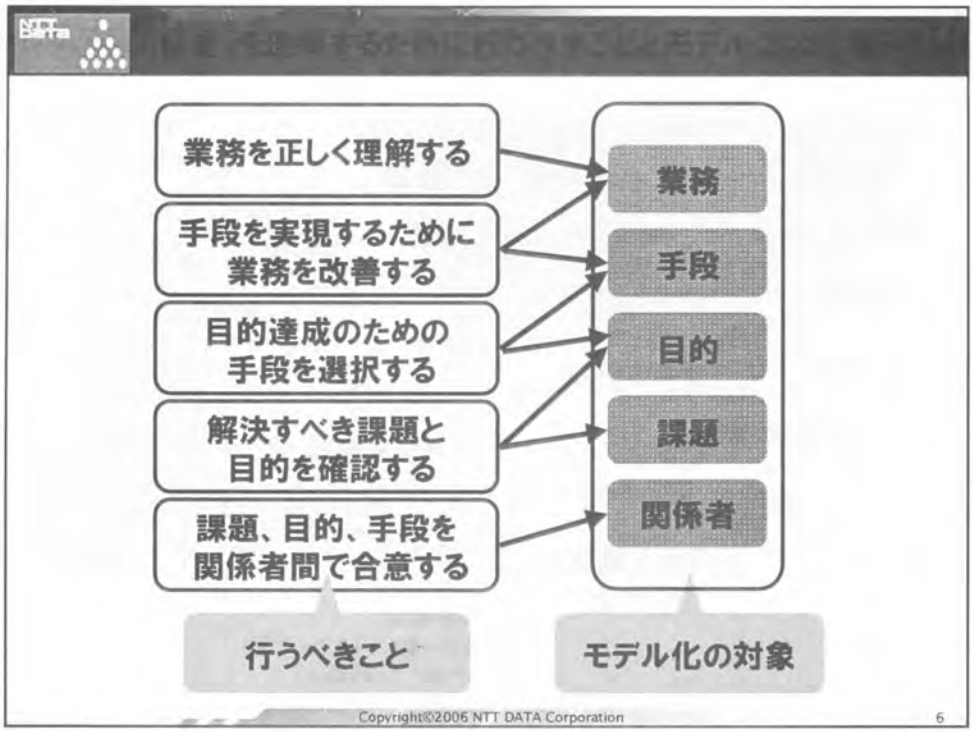
- 少ない記述量で十分に理解しあうためには？
  - 明示的に書いていないことを理解する
  - 要求仕様の「背景」を理解することで記述の不足を補う
  - 「業務を知る」「お客様を知る」

何を知れば、「仕様の背景」を十分に知ったことになるの？

「仕様の背景」は、どう表現(＝モデル化)し、お客様と確認しあえばよいの？


お客様と課題
システム の目的
業務 フロー
業務 機能
業務 プログラム
モジュール 設計
システム 設計
要素技術
課題
課題
課題
Java
J2EE
XML
RDB
ERP
Pentium
Giga-LAN
Copyright©2006 NTT DATA Corporation
4


お客様と課題
システム の目的
業務 フロー
業務 機能
業務 プログラム
モジュール 設計
システム 設計
要素技術
課題
課題
課題
「仕様の背景」 はこのあたり？
要求の 仕様化
設計・実装
Java
J2EE
XML
RDB
ERP
Pentium
Giga-LAN
Copyright©2006 NTT DATA Corporation
5

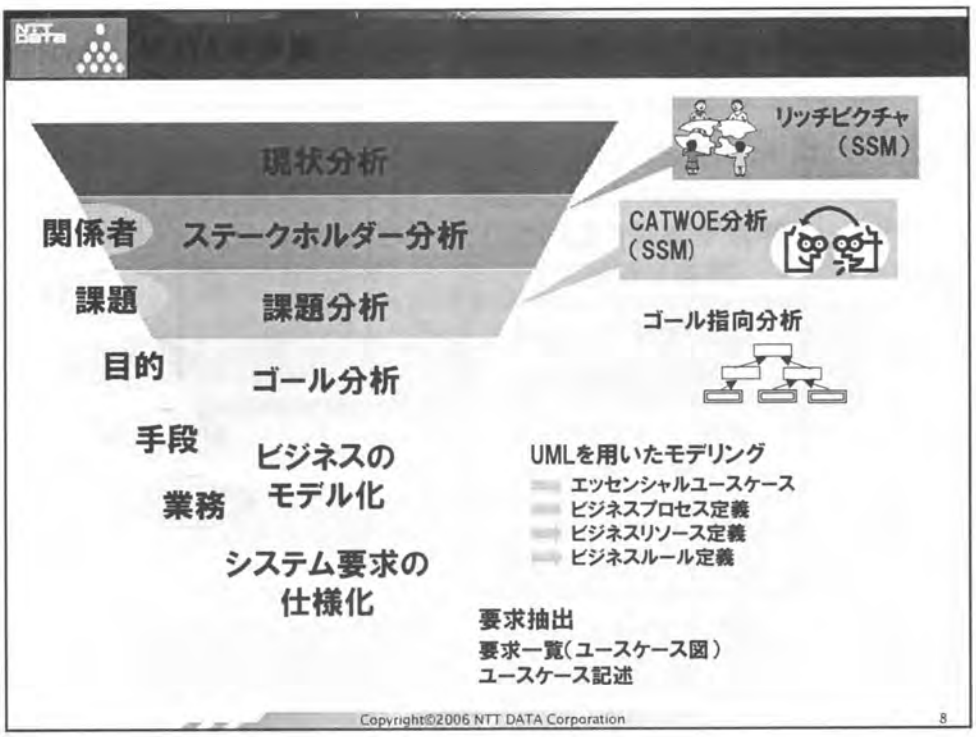


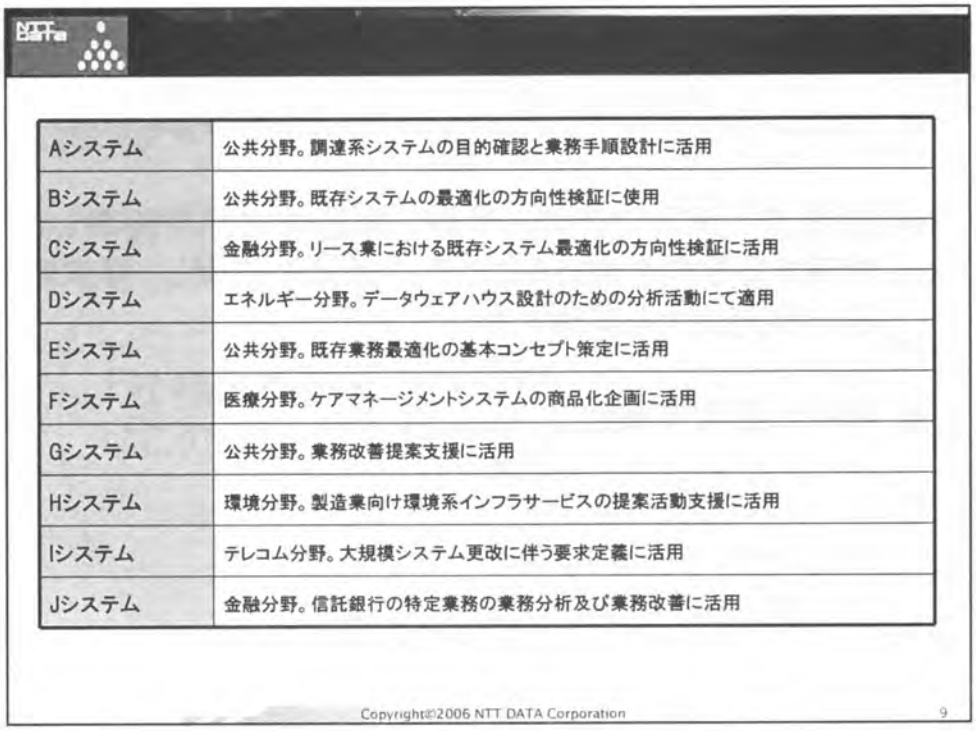
■ Model-Oriented Methodology for Your Awareness

■ **関係者、課題、目的、手段、業務をモデル表現し、吟味する手順を定めることで、「気付き」を促し、要求定義の質的向上を実現する**

■ **様々な手法の統合**

⇒ ソフトシステムズ方法論(SSM)
⇒ ゴール指向分析
⇒ エッセンシャルユースケース
⇒ UMLによる多視点モデリング





Aシステム	公共分野。調達系システムの目的確認と業務手順設計に活用
Bシステム	公共分野。既存システムの最適化の方向性検証に使用
Cシステム	金融分野。リース業における既存システム最適化の方向性検証に活用
Dシステム	エネルギー分野。データウェアハウス設計のための分析活動にて適用
Eシステム	公共分野。既存業務最適化の基本コンセプト策定に活用
Fシステム	医療分野。ケアマネージメントシステムの商品化企画に活用
Gシステム	公共分野。業務改善提案支援に活用
Hシステム	環境分野。製造業向け環境系インフラサービスの提案活動支援に活用
Iシステム	テレコム分野。大規模システム更改に伴う要求定義に活用
Jシステム	金融分野。信託銀行の特定業務の業務分析及び業務改善に活用

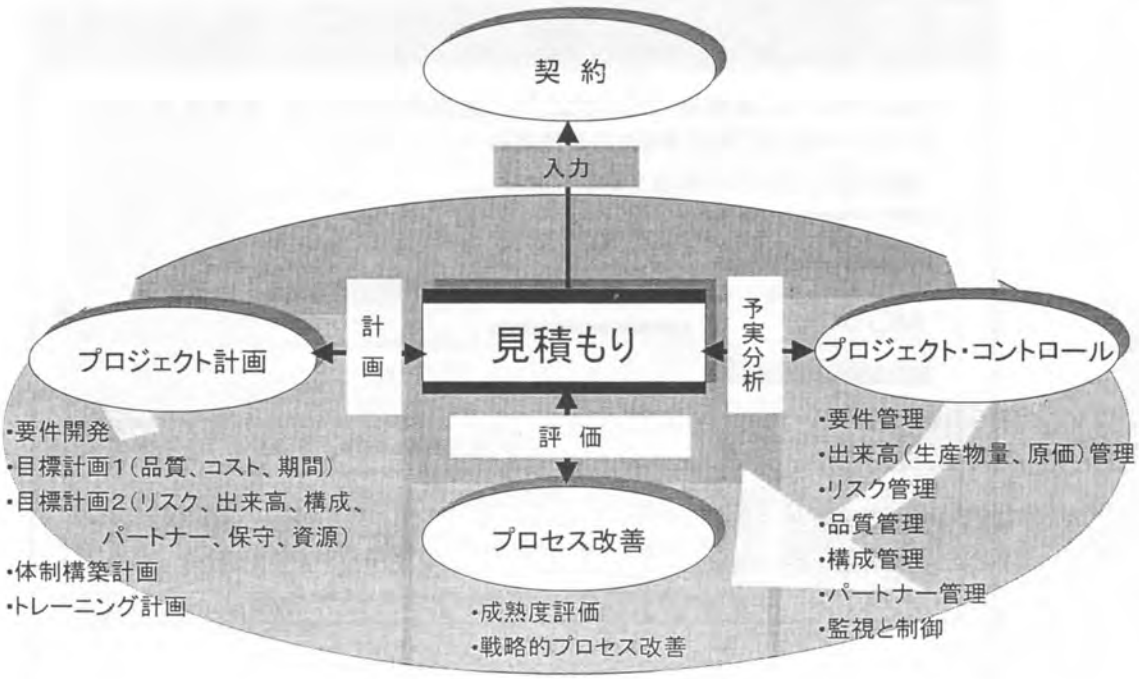
Kシステム	エネルギー分野。社会インフラ系サービスの改善活動に活用
Lシステム	環境分野。環境分野におけるビジネスの将来性に関する分析を実施
Mシステム	公共分野。提案書作成に当たって全国規模の調査業務における課題分析を実施
Nシステム	公共分野。省庁横断システムにおける、各省庁のビジネスプロセスの統一に活用
Oシステム	交通分野。新しいターミナル開設に伴うITシステムへの要求事項整理に活用
Pシステム	公共分野。提案書作成に当たって課題分析とシステム化目的の整理に活用
Qシステム	公共分野。全国に分散する拠点で使われるITシステムの業務プロセス分析に活用
Rシステム	環境分野。静脈物流におけるITシステムの可能性に関する分析に活用
Sシステム	金融分野。証券取引所におけるビジネスゴールとIT計画の分析に活用

- ガイドライン、事例集、ワークシート、適用のティップス、各種解説資料をNTTデータ社内で共有するための仕組み
- 実践的なノウハウの共有を可能にする

要件を定量的に捉えた見積りモデル
（要件：品質要件および開発環境要件）

2006年8月21日
株式会社ジャステック
太 田 忠 雄

1．1　生産管理に基づく見積もりモデルとは？



2．1　生産物量と生産性から成るコスト見積もり

【基本アルゴリズム】

●見積もりモデルの基本アルゴリズムは、生産物量見積り「'V_i'」および生産性見積り「'P_i'」から成立している。

●ある工程 i の生産物量をV_i、生産性をP_i、標準生産物量をV^B_i、生産性のベースラインをP^B_iで表現すると、コストC_iは次式で求まる.

$$C_i = V_i \times P_i = V^B_i (1+a_i) \times P^B_i (1+b_i)$$

Where, a_i: $(\Sigma \alpha_{ij})$ b_i: $(\Sigma \beta_{ij})$

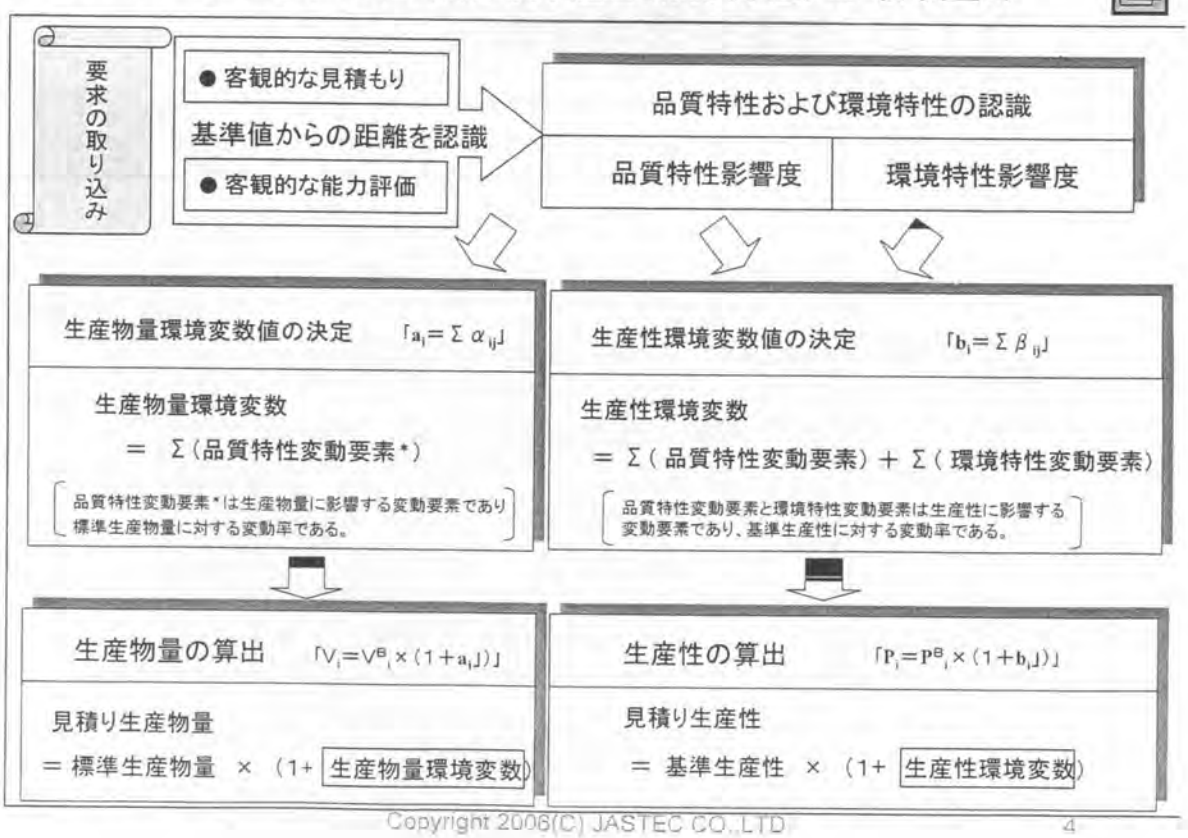
(注)'P_i' は 生産物単位のコストである。
('P_i'を生産物単位のスピードとする場合は、C_i は総工数になる)

【環境変数】

●a_i、b_iは、それぞれV^B_iおよびP^B_iに対して開発環境の違いや品質要求の多寡による変動を吸収する「環境変数」と呼ぶパラメータである（a_i:生産物量環境変数、b_i:生産性環境変数）。

●a_i、b_iは、品質特性と環境特性から影響される独立した変動要素（α_{ij}、β_{ij}）から構成している。

2．2　プロジェクト特性（品質特性と環境特性）の取り込み



3.1(1) 生産性環境変数(ユーザ責)

生産性見積り方式

$P_i = P^b_i \times (1 + \Sigma \beta_{ij})$　P_i：工程iの生産性　P^b_i：工程iの基準生産性　β_{ij}：工程iの生産性環境変数

1/4

特性タイプ	主特性	副特性	評価の観点（概略内容）	影響度（ユーザ責対象：%）			
				要件定義	設計	製作	テスト
環境特性	業務特性	業務ナレッジ	顧客の開発対象業務に対する業務ナレッジが生産性に及ぼす影響	-10 ～ 50	-10 ～ 10	-	-10 ～ 10
	ハードウェア特性	安定度/信頼度/使用度	システムもしくは製品となるハードウェアの安定度・信頼度	-	-5 ～ 5	-	-5 ～ 5
	ソフトウェア特性	安定度/信頼度/使用度	システム/製品となる他社作成ソフトウェアもしくはCotsの安定度・信頼度	-	-5 ～ 5	-	-5 ～ 5
	コミニュケーション特性	顧客窓口特性	意思決定能力（期限遵守、決定事項の履る度合）	-10 ～ 20	-10 ～ 10	-	-7 ～ 7
		工期の厳しさ	基準工期（月）＝2．7×（人月）$^{1/3}$に対し▲３０%限度とした短期化度合	0 ～ 10	0 ～ 10	0 ～ 5	0 ～ 7
		コミニュケーション基盤	開発拠点分散、資料等情報共有、電子媒体・システム具備など物理的基盤充実度	-10 ～ 10	-5 ～ 5	-3 ～ 3	-3 ～ 3
		レビュー体制	無駄なレビュー（重複多段階等）の排除およびレビュー効率向上への工夫度合	-5 ～ 5	-5 ～ 5	-3 ～ 5	-5 ～ 5

3.1(2) 生産性環境変数(ユーザ責)

2/4

特性タイプ	主特性	副特性	評価の観点（概略内容）	影響度（ユーザ責対象：%）			
				要件定義	設計	製作	テスト
環境特性	開発環境特性	開発手法/開発環境	開発手法・環境(ソフト/ハード/ツール)の信頼性、占有率などを考慮した使用実績	-3 ～ 3	-3 ～ 3	-5 ～ 5	-5 ～ 5
		テスト手順書水準	テスト手順の具体化度（操作手順＆入出力の具体化の要求水準）	-	-	-3 ～ 3	-3 ～ 3
	工程入力情報特性	業務関連資料	必要資料の具備状況（正確性、信頼性を含む）および使い易さ（検索性、理解性）	-10 ～ 10	-7 ～ 7	-3 ～ 3	-3 ～ 3
		他システム関連資料	必要資料の具備状況（正確性、信頼性を含む）および使い易さ（検索性、理解性）	-10 ～ 10			
		規約・標準化関連資料	必要資料の具備状況（正確性、信頼性を含む）および使い易さ（検索性、理解性）	-7 ～ 7			
	顧客の協力特性	役割分担特性	顧客がベンダに協力する度合および顧客とベンダとの役割分担の明確性	-10 ～ 20	0 ～ 10	-	0 ～ 10

3. 1(3)生産性環境変数(ユーザ責)

特性タイプ	主特性	副特性	評価の観点（概略内容）	影響度（ユーザ責対象：%）			
				要件定義	設計	製作	テスト
現行資産特性	既存母体調査ツール機能調査		調査ツール機能（絞込み、モニタリング、リバース等）の数	-	-20 ～ 15	-10 ～ 5	-20 ～ 10
	母体資産の具備状態	製品設計書有無メンテナンス状態	母体システムの設計書有無および設計書が存在した場合のメンテナンス状態が、改造作業に及ぼす影響を評価	0 ～ 10	0 ～ 30	0 ～ 30	0 ～ 25
		既存テスト環境流用水準	既存のテストリソースの流用度合	-	-	-20 ～ 0	-30 ～ 0
	現行システムの品質	正確性	現行システムが正しく動作しない場合の生産性に及ぼす影響を評価	0 ～ 15	0 ～ 25	0 ～ 25	0 ～ 25
		解析性	現行システムの改造箇所特定（改造設計）におけるドキュメント、ソースコードの解析容易性を評価	0 ～ 10	0 ～ 25	0 ～ 25	0 ～ 15
		環境適用性	現行システム資産を別環境への移植し改造する開発における、別環境への適用容易性を評価	0 ～ 10	0 ～ 25	0 ～ 25	0 ～ 15

3. 1(4) 生産性環境変数(ユーザ責)

特性タイプ	主特性	副特性	評価の観点（概略内容）	影響度（ユーザ責対象：%）			
				要件定義	設計	製作	テスト
品質特性	効率性	実行効率性	実行効率に対する一般的要求水準（既知）の最適事例を基準にした要求水準	0 ～ 5	0 ～ 10	0 ～ 5	0 ～ 10
		資源効率性	資源効率に対する一般的要求水準（既知）の最適事例を基準にした要求水準	0 ～ 5	0 ～ 10	0 ～ 5	0 ～ 10
	保守性	解析性	ソースコードの解析性をコード化規約に定めるコメントに対する要求水準により評価	-	-	0 ～ 5	-
		安定性	ソフトウェア変更に対しシステム品質維持可能とする水準をライフサイクル目標年数の長さにより評価	0 ～ 5	-3 ～ 7	-3 ～ 5	-
	移植性	環境適用性	多様なハード、ソフト、運用環境に適用させる要求の水準	0 ～ 7	0 ～ 7	0 ～ 3	0 ～ 5
		移植作業性	環境を移す際に、必要な労力を低減させる要求の水準	0 ～ 7	0 ～ 7	0 ～ 3	0 ～ 5
		規格準拠性	移植性に関する国際/国内規格または規約を遵守する要求の水準	0 ～ 7	0 ～ 7	0 ～ 3	0 ～ 5
		置換性	使用環境/条件を変更せずに他のソフトウェア製品と置き換えて使用可能とする要求の水準	0 ～ 7	0 ～ 7	0 ～ 3	0 ～ 5

(注) 移行、教育、保守、運用作業は生産性環境変数の適用対象から除いている。

3. 2(1)規模環境変数(ユーザ責)

生産物量見積り方式

$V_i = V^b_i \times (1 + \Sigma \alpha_{ij})$　V_i：工程iの生産物量　V^b_i：工程iの標準生産物量　α_{ij}：工程iの生産物量環境変数

1/2

特性タイプ	主特性	副特性	評価の観点	影響度（ユーザ責対象：%）			
				要件定義	設計	製作	テスト
品質特性	機能性	合目的性	利用者/利害関係者の広がり、コンテンジェンシー対応、不正移行データ対応等の該当事象数	0～50	0～50	0～50	0～50
		正確性	正確性（検証）に関わる標準テスト密度を基準にしたテスト項目量への要求水準	-	-	0～20	0～50
		接続性	他システムとの接続によるコード変換、フォーマット変換数	0～5	0～20	0～20	0～20
		セキュリティ	対応が必要なセキュリティ実現機能数、但し機能要件に定義されている部分は除く	0～20	0～20	0～20	0～20
	信頼性	成熟性	故障低減に必要な実現機能数	0～5	0～10	0～10	0～10
		障害許容性	異常検知に必要な機能数	0～5	0～10	0～10	0～10
		回復性	再開処理に必要な実現機能数	0～5	0～10	0～10	0～10

3. 2(2)規模環境変数(ユーザ責)

2/2

特性タイプ	主特性	副特性	評価の観点（概略内容）	影響度（ユーザ責対象：%）			
				要件定義	設計	製作	テスト
品質特性	使用性	理解性	理解性向上（機能等）のためのプレゼンツール等の作成対象数	-	0～10	-	-
		習得性	習得性向上（使い方等）のためのマニュアル等の作成対象数	-	0～10	-	-
		操作性	操作性向上（心理的/肉体的配慮、運用やインストール容易性等）のための実現機能数	0～10	0～20	0～20	0～20
	保守性	解析性	解析に必要な実現機能数	-	0～10	0～10	0～10
		変更作業性	作成する保守用ドキュメントの数	-	0～10	-	-
		試験性	試験に必要な機能数	-	0～10	0～10	0～10
現行資産特性	現行資産品質	正確性	現行システムが正しく動作しない場合の現行保証（テスト量）に及ぼす影響を評価	-	0～5	0～15	0～20

（注）移行、教育、保守、運用作業は規模環境変数の適用対象から除いている。

3．3 取り得る値の範囲（ユーザ責）

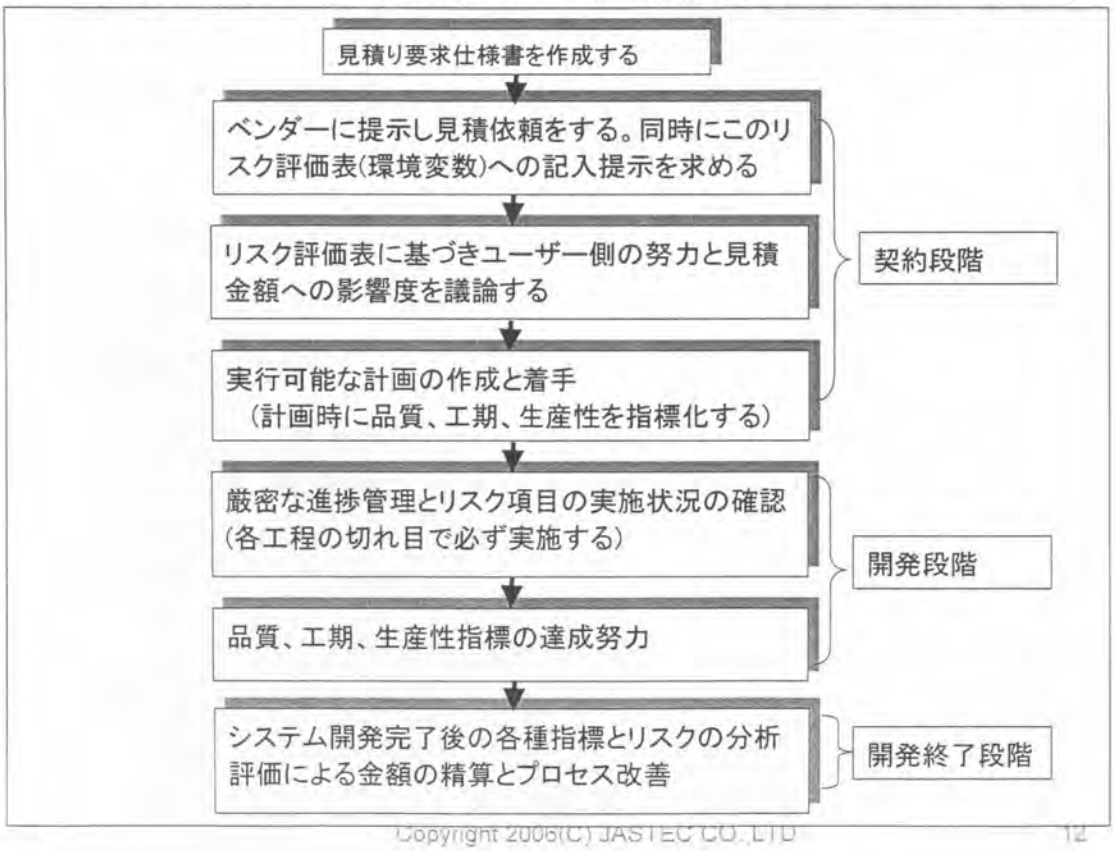
「ベンダー責および特異点を排除した値」

種別	特性		副特性	基設 本計	設 パッケージ計	作 プログラム成	テ 統 ス 合ト	テスト システム
生産物量環境変数	品質特性	機能性	正確性、他2項目	−20 ～ +20	−20 ～ +20	−15 ～ +15	−30 ～ +30	−30 ～ +30
		信頼性	成熟性、他2項目					
		使用性	操作性、他2項目					
		効率性	実行効率性					
		保守性	解析性、他2項目					
	仕様変更		変更棄却対象率	−5 ～ +25	−5 ～ +20	−5 ～ +15	−5 ～ +10	−5 ～ +10
			変更正味棄却率					
			変更追加率					
生産性環境変数	品質特性	機能性	合目的性、他3項目	−30 ～ +30	−20 ～ +20	−10 ～ +10	−10 ～ +10	−20 ～ +20
		効率性	実行効率性、他1項目					
		保守性	解析性、他1項目					
		移植性	環境適用性、他3項目					
	環境特性	業務特性	業務ナレッジ	−40 ～ +40	−30 ～ +30	−10 ～ +10	−20 ～ +20	−30 ～ +30
		ハードウェア特性	使用実績度					
		ソフトウェア特性	使用実績度					
		コミュニケーション特性	組織複合度、他4項目					
		入力情報特性	業務関連資料、他2項目					
		開発環境特性	開発手法・開発環境、他1項目					

（注1）当社標準工程定義と乖離がある場合、環境変数影響度が限界地を超える事があります。

（注2）当社は環境変数影響度が上記範囲内に収まるよう作業改善を推進致します。

3．4 環境変数「ユーザ責」（リスク評価表）として使用する場合



ご清聴ありがとうございました。

参加者アンケートのまとめ

以下は８月の Forum 参加者の方々を対象に行ったアンケートのまとめです．回答率はおよそ５０％でした．

Q1. 今回の Forum 全体の評価は？

□ 面白かった	15
□ まあまあ	9
□ つまらなかった	4

Q2. ５人のスピーカのプレゼンテーションは？

(1) 海谷さん：	□ 面白かった	4
	□ まあまあ	13
	□ つまらなかった	11
(2) 菊島さん：	□ 面白かった	15
	□ まあまあ	11
	□ つまらなかった	2
(3) 三瀬さん：	□ 面白かった	21
	□ まあまあ	6
	□ つまらなかった	1
(4) 斉藤さん：	□ 面白かった	10
	□ まあまあ	14
	□ つまらなかった	3
(5) 太田さん：	□ 面白かった	6
	□ まあまあ	17
	□ つまらなかった	4

Q3. パネル討論はどうでしたか？

□ 面白かった	11
□ まあまあ	11
□ つまらなかった	5

Q4. その他自由なコメントをお寄せください

コメント１： 「これからのシステム開発：要求工学の挑戦」というタイトルだったが、何に対して「挑戦」していたのだろうか？

コメント２： 太田さん、斉藤さんの話も１０分と言わずもっと聞きたいと思いました。

コメント３： 要求工学は、超上流フェーズから設計フェーズまで幅が広いため、対象とするフェーズを絞って議論しないと、焦点がぼけてしまうように感じました。また、実例と理論の

区別がつきませんでした。実例を基にした説明ならば、もう少しデータを用いた具体報告が聞けたら良いと思います。

コメント３：　とても有意義なお話を聞かせていただき、ありがとうございました。プレゼンの中でもありましたが、現状のシステム開発ではステークホルダーが多岐に渡っており、要求工学もそれぞれの立場・視点を考慮する必要があるかと思います。（今回のフォーラムではいろいろな視点でのお話があり、受講者の方で情報を取捨選択するスキルが必要だったと思います。）それぞれの立場でも活かされるような要求工学の発展を期待しています。

コメント４：　要求工学が扱う要求は、お客側の多様性が強い。そこに方法論を持ち込むと多少の多様性への制限が加わるが，その不便を越えるメリットを創造する方法論が望まれる。UML 等、方法論毎のメリット、デメリットの実証的説明があると聴衆は取捨選択できる

コメント５：　「要求を正しく定義して，RFP を作成する」観点からの議論が多かったのですが，「必要でないかもしれないシステムを正しく作る空しさ」をたまに感じるエンジニアとしては，「この要求自体は正しいのか」という観点からの議論がもっとあっても良かったと思います。また，よい「要求仕様」とはどういったものかについて，もっと掘り下げた議論があってもよかったかなと思います。テーマに対して時間が足りないように思いました。

コメント６：　パネル討論は、スピーカに自由にしゃべらせるのではなく，事前に用意したテーマで、司会者主導で進めたほうがよかったと思う。

コメント７：　要求工学の中で何がホットトピックなのか全容がつかめなかったが，現場のニーズとして要求工学へ期待が大きいことは確認することができた.

コメント８：　要求工学は、ソフトウェア工学もそうかもしれませんが、名前と期待が先行しているようです。あえて、改善点をを申し上げれば、以下のような Forum 全体の流れが最初に説明されるとよかったかとも思います。

１）問題は何で、

２）それに対するアプローチは、何が、どのようになされている

３）２）について、アカデミックな側面で有効なものがあるようには思えませんが、それでは、日々の問題解決をしながら仕事を進めなければならない現場では。いったい何がなされているか

コメント９：　要求工学というものの必要性を感じ、初めて参加させていただきました。そのせいかも知れませんが、工学的な部分があまりないことに、ちょっと落胆せざるを得なかったというのが正直なところです。また、現場の泥臭さと、学者の間のギャップを大きく感じ、それが要求工学進展の課題のようにも感じました。とはいえ、重要な取り組みであると考えており、機会があればまた参加させていただきたいと考えております。

コメント１０：　組み込み系のほうが体感できる話が多いですね。情報システム側からはひとの行動を予測しきれない。つまりは「非正常系」（三瀬さん）の現象が「ゼロ」になることはない，ということになる。そもそも、「ひと」と「情報システム」を並列に置くとか、「ひと」と「情報システム」を繋げることができる，と考えることが間違っているのかもしれない。

コメント１１：　内容にまとまりがなくひどかったと思う。MisUseCASE や超上流がどう要求工学に絡むのかの説明が、まるでなっていなかった。SEC の本は配るし、まるで SEC の宣伝の場でしたね。個々のスライドは、新しいバズワードの宝庫で（勉強不足ってこと;-p それなりに検索ネタとして新知識へのポインタ入手に使えるものでしたが、全体として、それがどう要求工学に位置づけるか？られるかが見えず、解説や議論もない(キャッチコピーからはこれが目的のはず)ので、怒って帰った人もいたそうです。まぁ、キャッチコピー自体が大風呂敷だったので、期待が大きい分不満も大きく、それでも何とか決着をつけようというの努力もなかったように思えました。なにせ、ほとんど質問もなく言いっぱなしに近い状態でした。SEA の常連の方々は、期待もほどほどだったと思いますが、一般参加の人たちは何を期待していたのでしょうね。SEC からのメールマガジンで知って来た人が多かったのでしょうが。。。SEC の実態がわかってよかったか。

コメント１２：　参加者の一部から看板に偽り有りだとの声が出ていたと聞いたが，残念ながら同感。パネルの司会者が「用意した話題は一つも出せなかった」と言っていたが，参加募集にうたわれた内容と実際のフォーラム内容の齟齬は，司会者の技量不足だけが原因ではないと感じた。従来からソフトウェア開発現場で皆が悶々としてきた諸問題・課題に対して，プレゼンタ・パネリストが，個人的経験と私見を好き勝手に言い合っただけに終わってしまった。「要求工学」なるものが，最上流プロセス付近の諸課題への散発的な取り組みの寄せ集めであって，「工学」はもちろん，「学」としてすら系統性（バックボーン）を持たない虚学であることを改めて実感して大変失望した。最上流プロセスは人間系との関わりが大きいので，単純に工学的なアプローチだけでは問題解決は無理であろうが，工学の分野で永年培われてきた「工学的な問題解決ノウハウ」の活用が，もう少し考えられないものなのだろうか？

コメント１３：　スピーカのタレント性はそれぞれ面白かったのですが、少々発散気味であったような気がします。使われないシステムや、欠陥による事故等の問題を考えると、受託側，開発側の観点からの議論だけではなく、依頼側，発注側（の社会的責任の観点）も含めた要件工学の研究が必要ではないかと思います。要求工学のフレームワークの再定義が必要かも知れないと思いました。

コメント１４：　時間不足のせいもあったが、すべての重要問題について、議論が生煮えだった。例えば、要件はたくさん書けばよいのか、少ないほうがよいの、とはいうものの、精粗のレベルは基準では改善できると思うのは幻想。核心は、システムエンジニアの能力だが、文章力のような基本的な能力さえ著しい低下傾向にあるし、まして、システム全体の把握のような能力を持つシステムエンジニアが枯渇しているのだから、状況は悪化する一方であろう。開発経験のない発注者が要件を書き、偽装請負の派遣要員がインプリメントしている現実では、要件の重要度評価など無理。

コメント１５：　要求獲得、定義の大切さはよくわかるのですが、具体的な個々の手法について，解説があったほうが「工学」としての存在意義がはｋっきりしたのではまいでしょうか。ただ，上流工程をいくら頑張っても、下流工程を協力会社にまかせておいたのでは，よいシス

テムは構築できないように思います。

コメント１６：　要求獲得とは、あいまいなものを明確にしていく作業だということには、だれも異論はないと思います。その割には、要求工学というあいまいなものを工学として位置づけていく作業、あるいは研究が希薄な感じがします。そういう意味で、要求工学に対する要求獲得を試みる必要があるのではないかと感じました。パネルですが、パネラーの要求工学に対する方向性がばらばらのまま終了したように見えました。ばらばらであるのはかまわないのですが、それぞれの意見の関連や相違について論じて、今後要求工学に対して何を目指していけばよいか、全体として一定の方向性が見えたらよかったと思います。

コメント１７：　工学的な内容が少し期待不足でした。パネル・ディスカッションでは、話のポイントが随分ずれていたように思いました。配布資料が間に合わなかったなど、準備不足が感じられました。

コメント１８：　斉藤さん、太田さんの発表を、もう少し聞きたかった。せっかく登場していただいているのにもったいない。大田さんの見積もり技法は、要求工学の観点からは、少し無理があると考えます。生産物量と生産性でのコスト見積もりは、もう少し仕様確定が行われた後であるならば、納得がいきます。詳細説明を全て聞いていないので、なんともいえませんが。私は、上流工程では、ＮＥＳＭＡで開発されたＦＰ概算法を採用していますが、この手法でも、顧客要求を要件に分解して、計測する単位に持ってくるのが、一つのポイントだと考えています。このあたりを今回聞く機会があればと思ったのですが、少し思惑がはずれました。斉藤さんのＭＯＹＡは、モデルを統合していいとこ取りをしているようなので、具体的に聞きたかった報告です。三瀬さんの発表は、組込みですが、市場問題などの品質管理実績から裏打ちされた内容だったので、説得力がありました。非正常系の要求を考慮する上では貴重だと考えられます。ただ顧客価値を考えた要求を創造する観点では、ＭＯＹＡのようなもので、ＲＦＰを作成したあとで考慮するようなもののように思えます。海谷さんのMisuse Cases Security は、確かに興味を引きます。要求事項に応じたセキュリティ面の概念の対応を考えるというのは、おもしろいと考えます。全体評価はまあまあです．というのは、要求時点での見積りをどのようにするのかに興味があったのですが、イマイチあてが外れたような感じだったからです。大田さんの発表や内容が問題といっているわけではありません。むしろ細かい環境変数などは、企業内で分析して、自分たちの生産性係数を考える参考資料になるのではないかと考えます。

コメント１９：　開催案内に「日本における要求工学の強み弱み、課題の取り組みについて議論する」とあったのに，特にピックアップされた議論もなくて残念でした。今回は、工学という観点の扱いが低かったように思います。工学視点での最近のトレンドやその取り組み事例の紹介（大学，企業），そこにおける課題など扱って欲しかったです。また，パネル・ディスカッションの内容がまとまりがなく，最終的にどういう落としどころだったかも不明です。次回に期待します。

コメント２０：　他社のお話をおうかがいできたので有意義でした。

コメント２１」　SEA Forum は初めての参加でしたが、とても有意義でした。豊富な実務経験のあるスピーカーのお話は、我々企業の現場で実務を担当しているものの胸に響くものがありました。

コメント２２：　超一流企業でも、私の部門と同じ問題意識、課題をもってること、自分たちの課題解決方向が間違っていないことが確かめれれてよかった。パネル討論で、あまり抽象的な話をされても意味が分かりません。アナロジーとは一体なんだったのでしょうか？「システムが人間の体で言うと血液」？？

コメント２３：　要求工学なのになぜ見積もりの話が入ってくるの？ しかもスピーカーはでしゃばり過ぎ。パネルでは海谷さんにもっと要求開発について発言して欲しかった。
MOYA については，背景ではなくそのものを聞きたかった。

コメント２４：　パネルのテーマもしくは趣旨を明確にしてから議論したほうが発散を防止できるように思います。

コメント２５：　パネル・ディスカッションの最後に、中谷さんから「金庫問題」を例に提唱された問題、つまり、情報システムが現実社会にどのような価値・サービスを提供していけるかという点について、もう少しスピーカのみなさんのご意見を伺ってみたかったです。

コメント２７：　私の回りで行われているソフトウェア開発は、その大半が 2 次請けであり、直接要件をコントロールしていくことが難しいケースが多々あります。業界内にはこういった要員が数多くおり、そういった立場の視線で議論をしても面白いのではないかと思います。

コメント２８：　今回，初めて SEA の Forum に参加しましたが、非常に刺激を受けました。機会が有れば、また参加したいと思います。SEA の活動にも興味を持ったのですが、QWb に載っている機関紙 SEAMAIL の目次があまりに古すぎます。その他のページの情報も更新が遅れているようで、新規会員の参加は余り望んでいないのかという印象を受けます。ボランティア組織で、幹事の方々の負担もおありでしょうが、これだけ充実した「場」であるのに、少し残念に思います。

SEA年次総会報告

事務局

今年度の SEA 総会は SEA Forum June (2006年 6月 16日) の夕刻, Forum の会場である JJK 会館で開催された.

事務局から報告し承認された昨年度の収支および新年度の予算は次の通りである：

収支計算書 (2005年 4月 1日～2006年 3月 31日)

支出の部		収入の部	
人件費	3,630	新入会費	110,000
事務所費	2,302,272	更新会費	2,176,000
印刷費	845,274	賛助会費	2,200,000
通信費	995,330	雑収入	72
会議費	208,615		
支部支援費	397,460		
国際活動費	45,420		
消耗品費	65,404		
雑 費	14,385		
当期収支差額	－391,718		
合 計	4,486,072	合 計	4,486,072

予算案 (2004年 4月 1日～2005年 3月 31日)

支出の部		収入の部	
人件費	20,000	新入会費	330,000
事務所費	2,400,000	更新会費	2,400,000
印刷費	1,000,000	賛助会費	2,-00,000
通信費	1,000,000	EVENT収入	200,000
会議費	130,000	雑収入	70,000
支部支援費	240,000		
国際活動費	50,000		
消耗品費	80,000		
雑 費	30,000		
当期収支差額	0		
合 計	5,000,000	合 計	5,000,000

編集後記

☆

ちょっと分厚いVol.1, No.1-3 合併号をお届けします.

☆☆

この号は当初，7月に開催されたSS2006 特集にする予定だったのですが，6月＆8月のSEA Forum の報告資料んpボリュームが多くなってしまったので，SS の報告は次号に回すことにしました.

☆☆☆

冒頭のレポートはSEA 幹事の1人である野中哲さんが3月にウィーンで開かれたETAPS 会議併設の2つのワークショップに参加された報告です.

☆☆☆☆☆

次のレポートは，6月Forum「オブジェクト指向の理想と現実」について，このイベントを企画された幹事の石川雅彦さん・桜井麻里さんによる報告と，酒匂寛さんが当日使われたスライドのコピーです．もう1人の講師・青木淳さんのスライドは，著作件の関係があってここには載せられません．いま青木さんが雑誌に連載されているエッセイがそのうちに単行本になるそうなので，それまでお待ちください.

☆☆☆☆☆☆

3番目のレポートは，SS2006 での「要求工学ワークショップ」を踏まえて行われた8月Forum の報告，配付スライド，そして参加者アンケートのまとめです．かなりのページ数になりました．アンケートにもあるように，半日にいろいろな話題を詰め込んだので，参加された方々には(スピーカも含めて)いささか欲求不満が残ったようです．「いずれ来年の適当な時期にリベンジを」と，企画者の中谷さんがいっておられました．☆☆☆☆☆☆☆

なおGPL v3 をテーマにして，やはり満員の盛況だった7月のForum については，特にレポートはだしません．GPL v3 のDraft は，その後新しいVersion が公開されたようです．詳しくは次のWeb PAge を御覧ください；

http://gplv3.fsf.org/gpl-draft-2006-07-27.html

☆☆☆☆☆☆☆☆

ソフトウェア技術者協会
〒160-0004 東京都新宿区四谷3-12 丸正ビル5F
Tel:03-3356-1077　Fax:03-3356-1072
E-mail:sea@sea.or.jp
URL:http://www.sea.jp/